# SEALTOUCH™ GRAPHICS TERMINAL

Lucas Deeco

## Models ST3220, ST3225, ST3420

Lucas Deeco

# ST3220 Series Graphics Terminal APPLICATIONS MANUAL

Includes: ST3220
ST3225
ST3420

Lucas Deeco • 31047 Genstar Road • Hayward, CA 94544 • (510) 471-4700 • Fax (510) 489-3500

COPYRIGHT 1990-1993

LUCAS DEECO

The information contained in this manual is the exclusive property of Lucas Deeco and shall not be reproduced, in whole or in part, without the written permission of Lucas Deeco.
Specifications are subject to change at any time without notice.

## WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

SealTouch and DeeCO are registered trademarks of Lucas Deeco .
VT320, VT220, VT100, VT52 et al are trademarks of Digital Equipment Corporation.

## LIMITED WARRANTY

Lucas Deeco warrants this product against defects in materials and workmanship for a period of one year (12 months) from the date of original shipment from the factory.

During this warranty period, Lucas Deeco will, at no cost to the Buyer, promptly repair or replace defective equipment returned to the factory, or other authorized warranty repair center, transportation charges pre-paid by the Buyer, and will return such equipment, transportation charges prepaid. Lucas Deeco's sole obligation shall be, at its option, to repair or replace any goods which have been determined to be defective by Lucas Deeco.

Equipment returned to the factory shall be accompanied by the following information:

- Returned Material Authorization (RMA) number, obtained from Lucas Deeco;
- reason for return, with a comprehensive description of the malfunction;
- the name and telephone number of the person to contact in the event of questions or problems; and,
- shipping instructions.

This warranty shall not apply to damage resulting from improper handling, accident, negligence, loss or damage in transit, or abuse (such as applying the wrong polarity or voltage power). This warranty shall be voided should the buyer attempt any repairs or alterations without prior written permission of Lucas Deeco.

Lucas Deeco makes no other warranty, either expressed or implied, and disclaims any warranty or merchantability or fitness for a particular purpose.

Any action by Buyer for any alleged breach of this warranty shall be brought to the attention of Lucas Deeco by the Buyer within the warranty period.

Repairs and/or replacement under the terms of this warranty SHALL NOT EXTEND THE WARRANTY LIFE OF THE ORIGINAL EQUIPMENT SUPPLIED.

## LIMITATIONS OF LIABILITY

THE BUYER AND LUCAS DEECO AGREE THAT THE SOLE AND EXCLUSIVE REMEDIES FOR BREACH OF ANY WARRANTY SHALL BE REPAIR OR REPLACEMENT OF DEFECTIVE PARTS ACCORDING TO THE TERMS DESCRIBED ABOVE. LUCAS DEECO SHALL NOT BE LIABLE FOR CONTINGENT OR CONSEQUENTIAL DAMAGES TO PERSONS OR PROPERTY, AND LUCAS DEECO'S SOLE LIABILITY IS AS SET FORTH ABOVE. THIS STATEMENT OF WARRANTY AND LIMITATION OF LIABILITY IS A COMPLETE AND EXCLUSIVE STATEMENT OF ALL WARRANTY AND LIABILITY REPRESENTATIONS OF LUCAS DEECO. It may not be varied, supplemented, qualified or interpreted by any prior dealings between the parties or by any usage of the trade or upon the face or reverse of any form to which this is attached or part of, nor may it be modified by any agent, employee, or representative of Lucas Deeco.

# CONTENTS


**SECTION 1 INSTALLING YOUR ST3220 ..... 1**

1 Procedures ..... 1
- 1.1 Unpacking ..... 1
- 1.2 Connecting The ST3220 To Your Computer/Modem ..... 1
- 1.3 Connecting A Printer ..... 1
- 1.4 Connecting A Keyboard ..... 1
- 1.5 Applying Power ..... 1

**SECTION 2 CONFIGURING YOUR ST3220 ..... 3**

2 Overview ..... 3
- 2.1 Entering And Leaving Set-Up ..... 3
- 2.2 Set-Up Menus ..... 3
  - 2.2.1 General ..... 3
  - 2.2.2 Display ..... 4
  - 2.2.3 Printer ..... 4
  - 2.2.4 Keyboard ..... 4
  - 2.2.5 Communications ..... 4
  - 2.2.6 Touch ..... 5

**SECTION 3 THEORY OF OPERATION ..... 7**
- 3.1 ANSI 3.64 Description ..... 7
- 3.2 Text Terminal Operation ..... 8
- 3.3 Graphic Terminal Operation ..... 9
- 3.4 Screen Viewports ..... 10
- 3.5 Drawing Objects ..... 13
  - 3.5.1 Erasing Objects ..... 14
- 3.6 Graphic Text ..... 14
- 3.7 Polygon Fill ..... 15
- 3.8 Fast Vector Mode ..... 15

**SECTION 4 COMMAND DESCRIPTIONS ..... 17**
- 4.1 Introduction ..... 17
- 4.2 Definitions ..... 17
- 4.3 Command Summary ..... 18
  - 4.3.1 Supported VT100/VT220 Commands ..... 18
  - 4.3.2 DeeCO Extension to VT100/VT220 Commands ..... 19
- 4.4 Detailed Command Descriptions ..... 21

**SECTION 5 TOUCH CAPABILITIES ..... 33**
- 5.1 Introduction ..... 33
  - 5.1.1 Capabilities ..... 33
- 5.2 Specifications ..... 33
- 5.3 Maintenance ..... 33
- 5.4 Software Commands ..... 34
  - 5.4.1 Touch Modes ..... 34
  - 5.4.2 Touch Reports ..... 35
  - 5.4.3 Touch Inquiry ..... 36
- 5.5 Button Commands ..... 36
  - 5.5.1 Define Button Location ..... 37
  - 5.5.2 Define Button Response ..... 37

SECTION 5 TOUCH CAPABILITIES (cont.)
5.5.3 Multi-State Buttons ..... 39
5.5.3.1 Behavior of Multi-State Buttons ..... 39
5.5.4 Button Manipulation Commands ..... 39
5.5.5 Button Modes ..... 40
5.5.6 Drawing and Erasing Buttons ..... 41
5.5.7 Button Menu Pages ..... 41
5.5.8 Button Resource Inquiry ..... 42
5.5.9 Pop-up Menus ..... 43
5.5.10 Button Time-Out ..... 44
5.6 On-Screen Keyboard ..... 44
5.6.1 Keyboard Modes ..... 44
5.7 Touch Protect ..... 45
5.8 Self-Check Functions ..... 45
5.8.1 Configuration Inquiry ..... 46

SECTION 6 RS-485 MULTI-DROP PROTOCOL ..... 47
6.1 Description ..... 47
6.1.1 Set-Up Fields ..... 47
6.1.2 Host Commands ..... 47

APPENDIX A – TERMINAL SPECIFICATIONS ..... 49

APPENDIX B – CONNECTOR PIN ASSIGNMENTS ..... 51

APPENDIX C – MECHANICAL DRAWINGS ..... 53

APPENDIX D – DEFAULT SETTINGS ..... 55

APPENDIX E – COMMAND COMPARISON ..... 57

APPENDIX F – COMMAND SUMMARY ..... 61

APPENDIX G – RETAINED GRAPHIC FRAME STORAGE ..... 65

APPENDIX H – RS232C ..... 67

APPENDIX I – KEYBOARD ..... 71

APPENDIX J – POWER UP MESSAGES ..... 73

APPENDIX K – DOCUMENTATION, ORDERING AND CUSTOMER SUPPORT ..... 75

APPENDIX L – ST3225 SPECIFICATIONS ..... 77

APPENDIX M – INSTALLATION PROCEDURES FOR CONDUIT COVERS ..... 79

APPLICATION NOTES ..... 81

INDEX ..... 89

# SECTION 1
# INSTALLING YOUR ST3220

## 1. PROCEDURES

EACH OF THE FOLLOWING STEPS SHOULD BE PERFORMED ON ALL NEW UNITS.

IT IS ADVISABLE TO RUN THROUGH THESE STEPS BEFORE CONTACTING LUCAS DEECO CORPORATION WITH A PROBLEM (THE FIRST THING WE WILL ASK YOU TO DO IS TO PERFORM THEM). FOLLOWING THESE PROCEDURES WILL SAVE YOU TIME AND GET YOUR TERMINAL UP AND RUNNING MORE QUICKLY!

**1.1 UNPACKING:** As you unpack, make sure to check for any visible damage to the terminal. Contact shipper if any damage is found. The contents of the shipping container should include: the SealTouch Terminal (ST3220), power cord and manual.

**1.2 CONNECTING THE ST3220 TO YOUR COMPUTER/MODEM:** Connect the ST3220 to your host computer or modem via the 25-pin connector on the back of the terminal using the RS-232, RS-422, or RS-485 (see Appendix B for the proper signal connections required). Once you have a correct cable connected between the terminal and your host computer you can proceed.

**1.3 CONNECTING A PRINTER:** If you are using this feature, connect an auxiliary printer to the 9-pin printer port on the ST3220 in the same manner as connecting to your host computer (see Appendix B for the proper signal connections required).

**1.4 CONNECTING A KEYBOARD:** A special port is available on the controller in which to connect an IBM PC AT signal compatible keyboard. Scan codes received from the keyboard are converted to ASCII code before transmission to your host or they may be sent directly, in the scan code format.

> ***IMPORTANT NOTE:*** ONLY IBM PC AT SIGNAL COMPATIBLE KEYBOARDS WILL FUNCTION PROPERLY.

DeeCO has added a special connector (Lemo RAE0.304NYL) to the keyboard which mates with the sealed connector on the back of the ST3220.

**1.5 APPLYING POWER:** First, connect the female end of the power cord to the ST3220. Then, connect the other (male) end of the power cord to any 120VAC 60HZ wall outlet. After a few seconds, the power indicator lamp (green LED) will light. At this point, the terminal is ready for operation. Do not make contact with the touch area of the screen before this time, as a touch may be interpreted as a failed I-R beam. The screen should read, "SELF TEST OK." If you hear a single beep, or multiple beeps, this indicates a failure and a message will be displayed for a short period of time which reads,

"SELF TEST FAILED [VALUE]" (where VALUE is a hexidecimal encoded number). Refer to the table below to decode that hexidecimal number, or series of beeps.

| Beeps | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | X | X | 1 | 1 | 1 | 1 | 1 | 1 |

Checksum(ROM)
Program RAM
IR System
Video RAM
Character Generator
Serial I/O
Not Used
Not Used

# SECTION 2
# CONFIGURING YOUR ST3220

## 2. OVERVIEW

This section describes the terminal set-up screens and how to use them. These seven screens allow you to examine or change terminal operating features such as transmit/receive speeds, parity, start bits, stop bits, screen background and so on.

## 2.1 ENTERING AND LEAVING SET-UP

The set-up menus, or screens, are accessible from either the optional keyboard or the on-screen keyboard.

If you are using the optional hardware keyboard use the 'SysRq' key in the top row to enter set-up. This key has a toggle-type operation and hitting it while in set-up will cause an exit from set-up.

If you are not using the optional keyboard you must get into the set-up screens via the touch keyboard. To bring the keyboard onto the screen, touch the bottom right-hand corner of the display are and a keyboard will appear (provided you have not already disabled keyboards by a host computer command). You get into the Set-Up menus by touching the "Set-Up" button on the function pad keyboard. If the keyboard that appears is either an ABC-type or QWERTY-type, touch the top right hand button labeled "Pad". The function pad keyboard will appear.

Once your set-up is complete, touch the "Exit" button, in the set-up screen, near the arrow buttons.

## 2.2 SET-UP MENUS

Once in the set-up screens, the main menu has six choices, GENERAL, DISPLAY, PRINTER, KEYBOARD, COMMUNICATIONS, and TOUCH. The arrow above the main headings show you which group is selected. The GENERAL selections are selected upon entering set-ups. The sections below the Main Menu headings can be changed by using the UP/DOWN arrow to position the pointer to the desired item and touching the "Enter" key on the numeric pad or the one on the screen.

### 2.2.1 GENERAL:

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| CLEAR DISPLAY | ACTION | Clears the text screen and returns the cursor home. |
| RECALL STATE | ACTION | Sets all features to their saved values. |
| SAVE STATE | ACTION | Saves all current feature settings. |
| DEFAULT STATE | ACTION | Returns all features to their factory default settings. |
| ON/OFF LINE | ON | On-line, communicates with host. |
| | OFF | Off-line, sends keyboard data to screen. |

### 2.2.2 DISPLAY:

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| AUTO-WRAP | OFF | Causes characters received after the right margin to be overwritten into the last character position of the current line. |
| | ON | Causes characters received after the right margin to automatically appear in first character position of the next line. |
| CURSOR ON | ON | Displays text cursor. |
| | OFF | Text cursor is not displayed. |
| CURSOR STYLE | BLOCK | Displays block-style cursor. |
| | UNDERLINE | Displays underline-style cursor. |
| SCREEN SAVER | ON | Programmable screen saver enabled (see SCREEN SAVER TIME-OUT). |
| | OFF | Screen saver disabled. |

### 2.2.3 PRINTER:

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| EXTENT | SCREEN | Print screen commands will print the full screen. |
| | SCROLL | Print screen commands will only print the scroll area. |
| TERMINATOR | NONE | No form feed sent at the end of a print page operation. |
| | FORM FEED | Form feed sent at the end of a print page operation. |
| PR TO HOST | NO | Printer may send data to host. |
| | YES | Printer may not send data to host. |

### 2.2.4 KEYBOARD:

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| AUTO REPEAT | ON | Pressing a key sends the character repeatedly until released. |
| | OFF | Pressing a key sends the character only once. |
| KEY MAPPING | ON/OFF | Mapping will ASCII encode all keys. Key mapping off will allow scan codes to pass to host. |
| NEW LINE | NO | The RETURN key sends a CR only. |
| | YES | The RETURN key sends a CR and LF. Received LF also performs CR. |

### 2.2.5 COMMUNICATIONS:

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| NETWORK | OFF | Networking inactive. |
| | ON | Networking active. When networking is active, the terminal ceases to respond to received data or transmit data such as key depressions. Certain commands received will enable or disable the terminal as a LISTENER (able to respond to received data), or a TALKER (able to transmit data). The network commands use the ID to designate a particular terminal as a talker or a listener. The network commands are always monitored, even if the terminal is not designated as a listener. See the description of these commands for further details. |
| THIS ID | ENTER ID | To enter a network ID, press "ENTER." The ID field will go to normal video. Enter the desired ID and press "ENTER." |
| NEXT ID | ENTER ID | To enter a network ID, press "ENTER." The ID field will go to normal video. Enter the desired ID and press "ENTER." |

## 2.2.5 COMMUNICATIONS (cont.):

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| BAUD | 75 | Set communication port baud rate. |
| | 110 | |
| | 150 | |
| | 300 | |
| | 600 | |
| | 1200 | |
| | 2400 | |
| | 4800 | |
| | 9600 | |
| | 19200 | |
| | 38400 | |
| DATA BITS | 8 | Set communications port for 8-bit data. |
| | 7 | Set communications port for 7-bit data. |
| PARITY | NONE | No parity sent or checked. |
| | EVEN | Even parity sent and checked. |
| | ODD | Odd parity sent and checked. |
| STOP BITS | 1 | 1 stop bit. |
| | 2 | 2 stop bits. |
| XON/XOFF | YES | Sends XOFF when input buffer reaches 200 characters. |
| | NO | Disables automatic XON/XOFF flow control. |
| IO TYPE | RS232 | Selects IO type to be RS-232C, data lines only. |
| | RS485, 2 CHNL | Selects IO type to be RS-485, 2 channel. |
| | RS485, 1 CHNL | Selects IO type to be RS-485, 1 channel. |

## 2.2.6 TOUCH:

| *FEATURE* | *SETTINGS* | *DESCRIPTION* |
|---|---|---|
| X-Y ENTER | OFF | X-Y enter mode disabled. |
| | ON | X-Y enter mode enabled. |
| X-Y EXIT | OFF | X-Y exit mode disabled. |
| | ON | X-Y exit mode enabled. |
| X-Y TRACK | OFF | X-Y track mode disabled. |
| | ON | X-Y track mode enabled. |
| X-Y HIGHLIGHT | ON | X-Y highlight mode enabled. |
| | OFF | X-Y highlight mode disabled. |
| X-Y BEEP | ON | X-Y beep mode enabled. |
| | OFF | X-Y beep mode disabled. |
| BUTTON ENTER | ON | Button enter mode enabled. |
| | OFF | Button enter mode disabled. |
| BUTTON HILGHT | ON | Button highlight mode enabled. |
| | OFF | Button highlight mode disabled. |
| BUTTON BEEP | ON | Button beep mode enabled. |
| | OFF | Button beep mode disabled. |

# SECTION 3
# THEORY OF OPERATION

## 3.1 ANSI 3.64 DESCRIPTION

The ANSI 3.64 standard describes a method of encoding information and commands for computer peripherals and other devices. The encoding is based on the 7-bit ASCII encoding defined by the ANSI standard 3.4 and the 8-bit extension to that encoding which defined the ANSI 3.41 standard.

A typical ANSI 3.64 command uses a "sequence introducer," "parameters" and a "final character" or "terminator," all encoded in ASCII. The sequence introducer signals the receiving device that a command follows. The parameters are usually decimal numbers required to perform the command. The final character determines which command the receiving device is to perform. A terminator is used when device-specific commands are used.

The ST3220 terminal uses the ESC, CSI and DCS sequence introducers and ST as a terminator. These ANSI-defined symbols are encoded as one to two characters. The two character versions must be used if 7-bit ASCII encoding is used between the host and the receiving device. The symbols are encoded as follows:

| ANSI SYMBOL | 8-BIT ENCODING | 7-BIT ENCODING | ASCII EQUIVALENT |
|---|---|---|---|
| ESC | 1BH | 1BH | ESC |
| CSI | 9BH | 1BH 5BH | ESC [ |
| DCS | 90H | 1BH 50H | ESC P |
| ST | 9CH | 1BH 5CH | ESC \ |
| – | 98H | 1BH 5BH 3CH | ESC [ < |
| – | 99H | 1BH 5BH 3EH | ESC [ > |

Parameters are ASCII encoded decimal numbers. The digits 0 through 9 are encoded as 30H through 39H in ASCII. Multiple parameters are separated by semi-colons (ASCII code 3BH). The ST3220 terminal extends the 3.64 to allow the specification of negative numbers. A negative number must have a colon (ASCII code 3AH) instead of a semi-colon separating it from the next parameter. The colon must be used even if there is no immediate next parameter.

Final characters have ASCII codes which fall between 40H and 7FH. Typically, a final character will be an alphabetic character, either upper or lower case.

Supplemental characters may appear after the introducer and before the first parameter which can alter the operation of the command. The ST3220 uses such supplemental characters to control non-ANSI features.

An example of an ANSI command sequence is the "cursor position command" used to move the alpha cursor to a particular row and column. The common definition of the command is:

CSI Pr ; Pc H

Where **Pr** stands for the decimal number of the row, and **Pc** stands for the decimal number of the column. The parameters are separated by a semi-colon. The command is introduced with the CSI introducer. It is terminated by the final character "H." The spaces shown in the definition are not part of the command, they are used to help illustrate the constituent parts of the command. This form of command definition is used throughout the manual to illustrate the general form of a command.

A specific instance of the "cursor position command" would replace the parameter symbols "Pr" and "Pc" with actual values. For example, to move the cursor to row 12, column 49, the following command form would be used:

CSI 12 ; 49 H

The actual information sent from the host to the terminal is the ASCII encoded command string. In the above example, the ASCII code for each character (except the spaces which are used for illustration) is sent to the terminal:

| COMMAND STRING CHARACTER/SYMBOL | ASCII CODE |
|---|---|
| CSI | 9BH (8-bit) or 1BH 5BH (7-bit) |
| 1 | 31H |
| 2 | 32H |
| ; | 3BH |
| 4 | 34H |
| 9 | 39H |
| H | 48H |

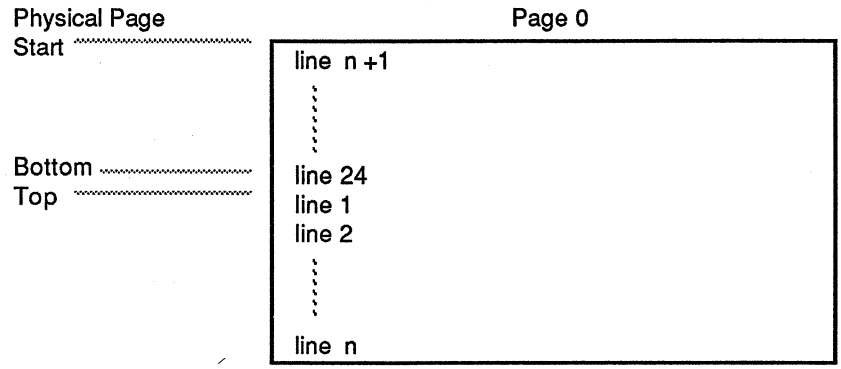
## 3.2 TEXT TERMINAL OPERATION

Text terminal operation simulates that of a DEC-style terminal. This terminal can display 1,920 characters arranged as 24 rows of 80 columns. New data sent to the terminal is displayed at the cursor location (a particular row and column), and may be manipulated on the screen by several commands which are based on the ANSI 3.64 standard.

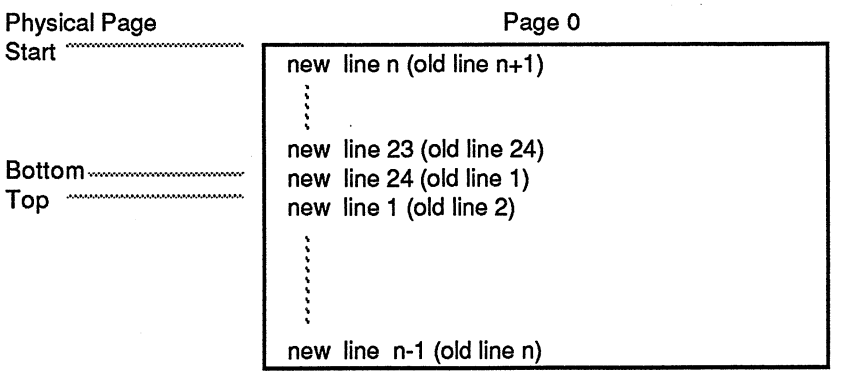
There are four character sizes. The size increases by multiples of two. The number of character rows and columns decrease by half for each increase in cell size: 24 rows by 80 columns, 12 rows by 40 columns, 6 rows by 20 columns, 3 rows by 10 columns.

The raster memory of the terminal consists of two pages, 0 and 1. A "page" is one full screen's worth of information.

The terminal mode of operation always uses raster page 0 for its information. The page may be thought of as a circular memory space in this mode, with its top and bottom border "wrapping around" (one and the same). Terminal functions which cause the whole screen to scroll, either up or down, merely change what is considered to be the "top-of-alpha-page." The top-of-alpha-page is a logical start, it is not the physical one.




**Before a Full Screen Scroll Up**



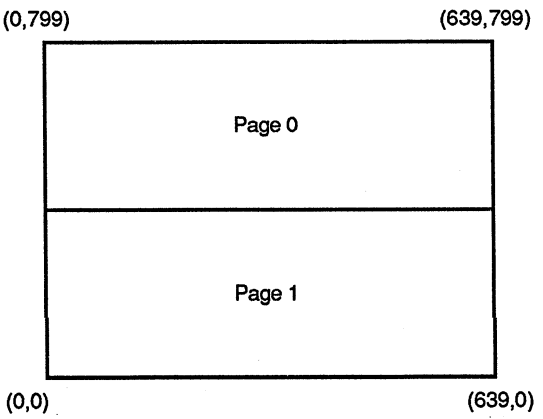

**After A Full Screen Scroll Up**

## 3.3 GRAPHIC TERMINAL OPERATION

The graphic terminal portion of the ST3220 terminal uses an extension of the ANSI 3.64 standard to encode its commands. There are commands included to draw objects such as vectors, circles, rectangles, and much more.

A graphic object is positioned within raster memory by specifying the coordinates of the object. The basic coordinate of the system, termed the "global coordinate system", assumes that raster memory is one large X-Y plane.

The dimensions of this plane are the screen width in the X direction by two times the screen height in the Y direction. For the 640x400 screen this would be 640x800.




Within this global system, a local coordinate system may be defined. This local coordinate system, known as the graphic window (or simply "window"), is defined by specifying a local origin in global coordinates, and a rectangular region relative to that origin. All object coordinates are assumed to be relative to the window origin. The rectangle represents boundaries at which the object may be "clipped."

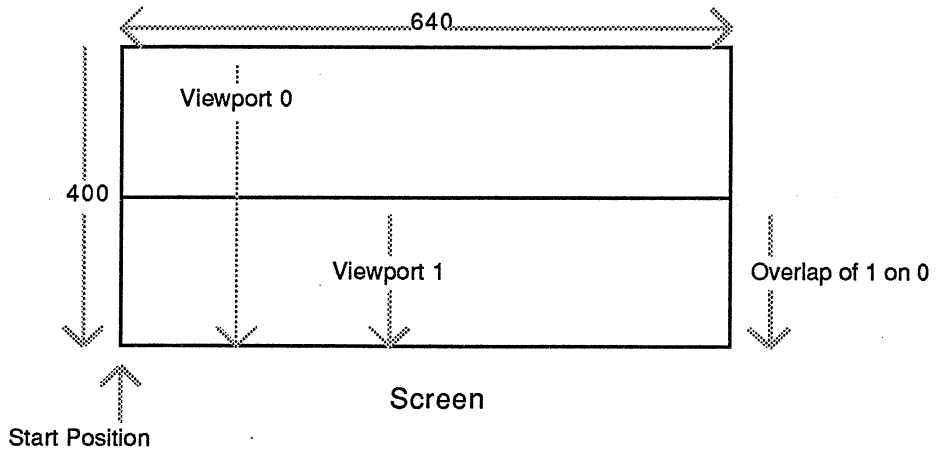
The window origin may be placed anywhere in raster memory, and the clipping rectangle may be of any size.

A simple, powerful mode of specifying the graphic window designates the window to be identical to the alpha (terminal page). The window origin is always the lower left corner of the page, and the clipping rectangle boundaries are at the page boundaries. This mode is known as, "alpha window."

## 3.4 SCREEN VIEWPORTS

All terminal information is stored in raster page 0. All graphic objects are drawn in the portion of raster memory encompassed by the graphic window. The portion of raster memory which is viewed on the screen is not necessarily dependent upon either the terminal page or the graphic window. The raster memory viewed on the screen is controlled by two user-definable "viewports."

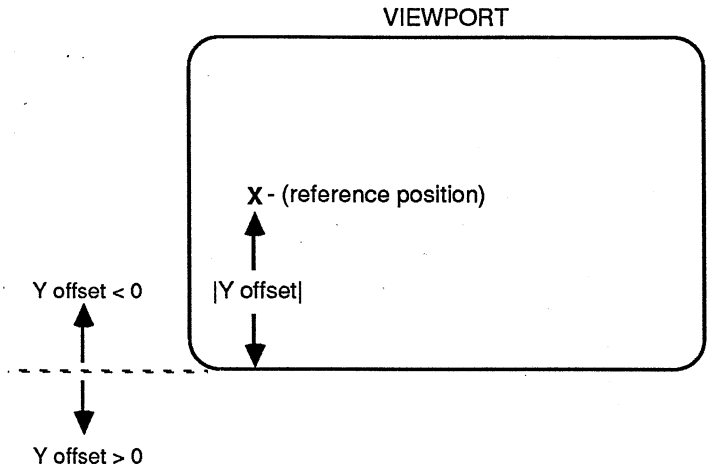
The two viewports, designated viewport 0 and viewport 1, are arranged on the screen with viewport 0 as the background and viewport 1 overlapping it. The size of viewport 1 is programmable from a size of zero (0) to a size encompassing the entire screen. The larger viewport 1 is, the more viewport 0 is obscured. The raster memory displayed in a viewport is defined by its "start position." The start position is the Y, X position, in global coordinates, of the lower left corner of the viewport. If this point is (Y0, X0) then the upper-right corner would be (Y)+viewport size-1, X0+screen width-1). For viewport 0, the size is always the screen height.



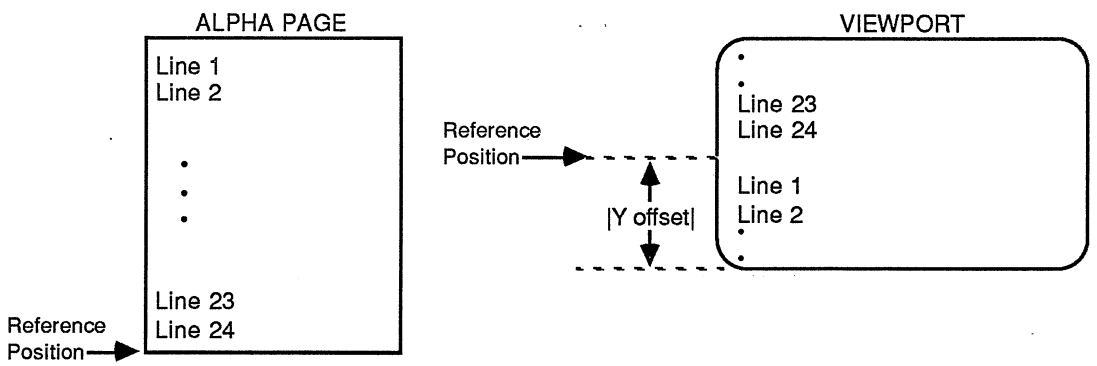

In the ST3220 terminal the start position X value must always be 0. This allows the display of a full, undistorted page width within the viewport.

There are four modes designating the start position of a viewport. Two modes relate to the graphic window and two relate to the alpha (terminal) page.

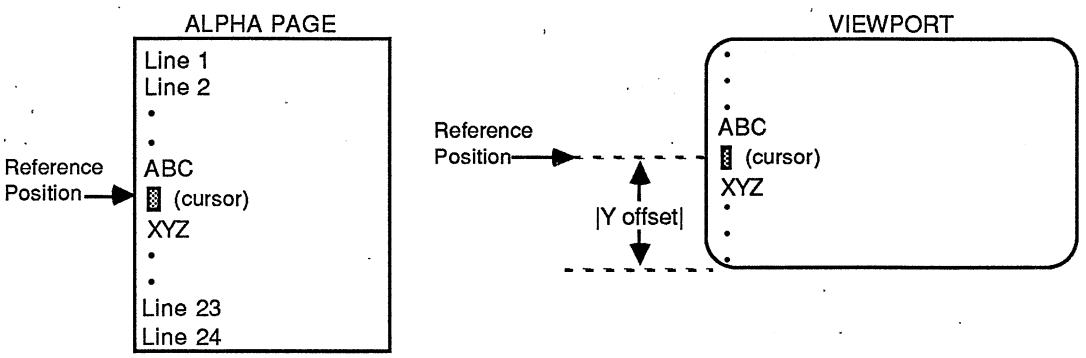
To display a portion (or all) of the alpha page, a viewport may be designated to start at either the current cursor row, or at some fixed row number (1 through 24). To fix the row relative to the cursor row is known as the "alpha relative" mode; to fix it relative to the bottom of the alpha page is the "alpha absolute" mode. The cursor row or bottom of alpha page are the "reference positions" for their respective modes. A Y offset (positive or negative) may be added to this reference position to obtain the start position of the viewport. Positive offsets move the reference position down (with respect to the viewport); negative offsets move it up.



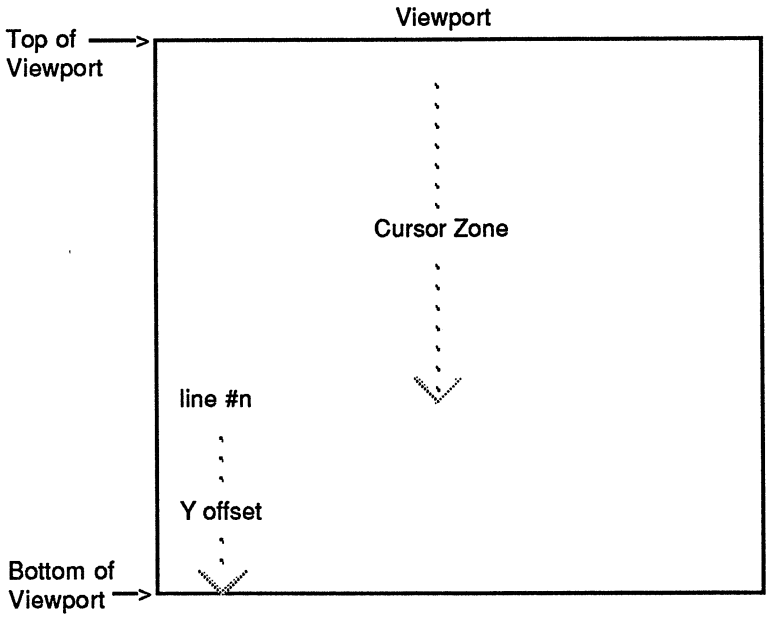

In alpha absolute mode, the reference position is the bottom of the alpha page. This position is kept fixed at a vertical distance of |Y offset| from the bottom of the viewport.



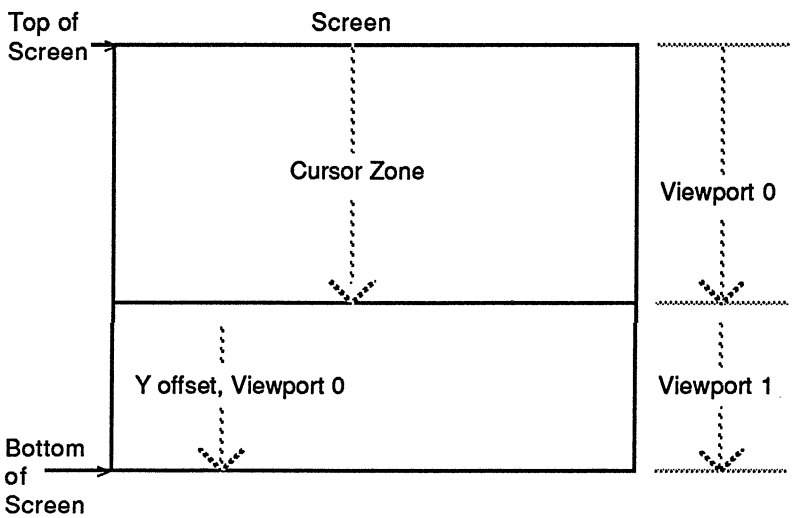

In alpha relative mode the reference position is the current cursor row. The cursor row is initially placed within the viewport a distance of |Y offset| from the bottom of the viewport.



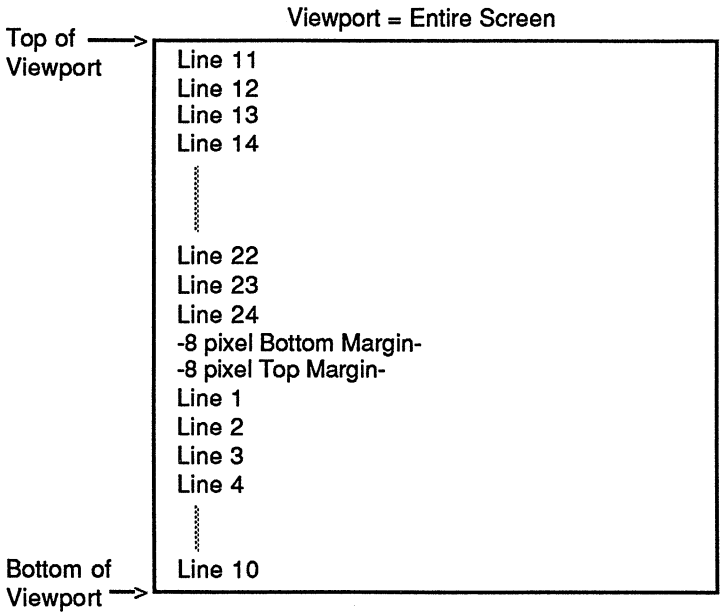

From then on, the cursor is always kept within the portion of the viewport defined "top" and "bottom" (by the top of the viewport and the bottom of the viewport), plus the Y offset.




The alpha relative mode is very useful when viewport 1 is overlapping viewport 0. Viewport 0 may be designated as alpha relative with a Y offset equal n magnitude to the size of viewport 1. This will keep the cursor in view at all times, preventing the situation where the cursor might move "behind" viewport 1.




The circular nature of the alpha page must be taken into account when displaying portions of the page within a viewport. This circular nature is envisioned by thinking of the page's top and bottom bounds as being the same bound. The bottom of the page "wraps-around" to the top of the page. The result of such a model is illustrated in the case of defining a viewport to be alpha absolute type, full screen size, and specifying line 10 as the reference line. The resulting screen image would have lines 1 through 10 at the bottom, and lines 11 through 24 at the top. The two vertical margins would be combined and would be visible between lines 24 and 1.




The graph viewport modes are termed "graph relative" and "graph absolute." In graph relative mode, the viewport reference position is the graphic window origin. The start position is automatically updated when the window origin is changed. In graph absolute mode, the reference position is the global origin and never changes.

As in the alpha viewport modes, a Y offset may be added to the reference start position to obtain the start position.

The displayed viewports, 0, 1, or both, allow split screen applications such as viewing terminal character rows on half the screen and drawing using graphics commands on the other half.

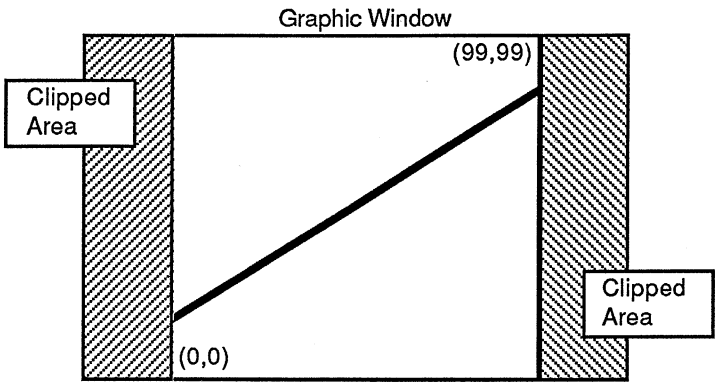
By selecting the graphic window to be identical to the alpha page, and using a viewport mode of graph relative or either alpha mode, the screen will display the alpha page and the graphic window all the time. All graphic objects will be drawn relative to the screen boundaries. This mode is known as the alpha window mode.

## 3.5 DRAWING OBJECTS

All terminal data is displayed in physical raster page 0. This data consists solely of displayable characters. Graphic objects may be drawn anywhere in raster memory, including the alpha page. This allows terminal text and graphics to be mixed or to remain separate.

The appearance of the characters may be altered by such attributes as reverse video, underlining, slanted font, or size. However they will always be drawn within the row and column designations of the alpha page.

Graphic objects are drawn relative to the window origin and may be clipped at the window boundaries. To "clip" an object is to draw only that portion of the object that lies within the clipping rectangle. Portions of the object lying outside the rectangle are not drawn, they are clipped at the rectangle boundaries.




Example of a Clipped Vector

**3.5.1 ERASING OBJECTS:** All objects, text and graphic, are drawn in raster memory by changing the contents of raster memory. The new data to be drawn typically replaces the old data already in the raster, thus over-writing whatever was there. There are 3 alternatives to this replacement operation, known as "complement", "set", and "clear" write modes. The write mode specifies a logical operation performed between the old raster data and the new raster data.

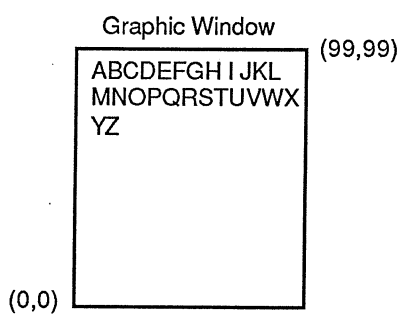
In complement write mode, the new and old data are exclusively "OR'ed" together. The result then replaces the old data. In set write mode, the old and new data are inclusively "OR'ed" together. Clear write mode combines the two data by "AND'ing" the old data with the logical complement of the new data.

The complement and clear write modes are particularly powerful in that they allow selective erasure of objects with the raster. For instance, if a vector (solid line) exists in raster memory, it can be erased by selecting complement or clear write mode, and then re-drawing the exact vector. The result is the erasure of the vector.

## 3.6 GRAPHIC TEXT

Text may be drawn within the graphic window as well as such objects as vectors, circles, and rectangles. Characters drawn using a graphic command are different from those drawn using the terminal mode. Graphic text may be started on any pixel, whereas alpha text is restricted to the character rows and columns of the terminal. Graphic text may be rotated on any 45 degree angle. And, graphic text may be confined, by clipping or auto-wrap, to the graphic window.

Restricting graphic text to the graphic window allows the window to operate like a "mini-terminal". It is not possible to scroll the window, but it still can operate much like a small terminal. To illustrate, take the example of a 100x100 graphic window and a text string consisting of the upper case alphabet. With auto-wrap on, and the cursor beginning in the upper left corner of the window, the resulting text string would appear like:



The standard character cell width is eight (8); the window is 100 pixels wide, so the window can accommodate 100/8, or 12, characters. The 13th would be clipped. Using auto-wrap, the 13th through 24th characters are placed on the next character line within the window. A character line is one standard cell height, or 16, pixels tall. Since the window is 100 pixels tall, 6 character lines would fit within it. The graphic window then behaves like a 6 row, 12 column terminal (without scroll). There is even a cursor much like the alpha cursor which can be displayed. Cursor movement commands like Line Feed and Carriage Return move the cursor within the window.

## 3.7 POLYGON FILL

Polygon fill refers to the feature of filling the interior of a polygon with a pattern. There are two major classes of polygons: 1) concave polygons, and 2) convex polygons. A subset of convex polygons, rectangles, comprises a third class.

Simple rectangle regions may be designated and filled with a variety of patterns. Such areas may even be designated to "flash" automatically. Such polygons are defined by the lower left and upper right corners of the rectangle. No "border" is required because the polygon is so simple.

Convex polygons other than rectangles may be defined by a continuous border (connected set of vertices) in raster memory. Such a polygon may be filled with any pattern available for rectangles. A convex polygon, simply defined, has no cut-outs and no indentations in its borders.

Concave polygons are defined by a continuous border similar to that of a convex polygon, but the border may indent or there may be cut-outs within the polygon. Such a polygon may be filled by a solid color (black or white). The color is the same color as the polygon's border.

## 3.8 FAST VECTOR MODE

The ST3220 terminal can be directed to enter a mode where its sole responsibility is to draw a connected set of vectors. The host must supply the vector end-point information. A vector is drawn from the previous end-point to the new end-point. This mode is known as the "Fast Vector Mode".

The terminal will acknowledge entering the Fast Vector Mode by sending the ASCII control code "ACK" (code 06H). The host must wait for this response before sending any vector information.

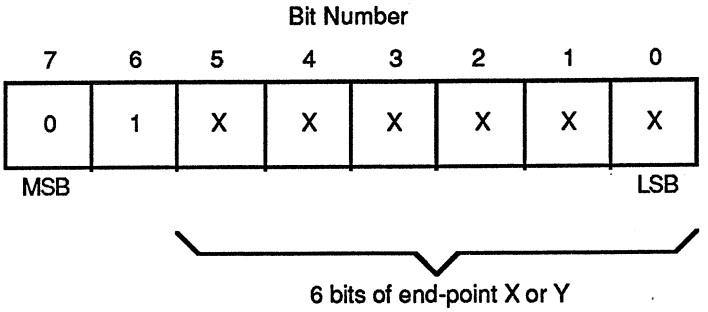
To draw a vector, the host must specify an Y, X coordinate. The terminal will respond by drawing a vector from the last point that the host specified in this mode, to the new point. The first point sent results in a vector drawn from the window origin to that point. The host may specify the line style with which the vector is drawn. Included in the selection of line styles is a "black" line style. This line style is invisible against a black background so it may be used to simply move to a new point without connecting it to the old point.

An end-point specification consists of one to six bytes of information. The first byte is known as the command byte. It specifies how many additional bytes will follow. The remaining bytes may or may not be sent by the host, depending on the command byte. These bytes specify two things: 1) line style, and 2) end-point.

If the line style is sent it immediately follows the command byte. It must be sent only if the line style is to change. The initial line style is solid white.

End-point information is encoded in four bytes. The Y and X components of the end point are 12-bit 2's complement integers. The 12 bits of information is split into two 6-bit values. These values are sent to the terminal in two separate bytes. The 6 bits of information is always adjusted toward the least significant bit within the byte. Bit number 6 of the byte is always set and bit 7 is always reset. Setting bit

6 prevents ASCII control codes from being sent inadvertently to the terminal. The four resulting bytes are termed High Y, Low Y, High X, Low X corresponding to which 6 bits the byte contains. The order of transmission is High Y, Low Y, High X, Low X. All bytes need not be sent. A byte must be sent only if the 6 bits of information it carries differs from the last end-point's corresponding 6 bits. All points must be specified in global coordinates. The initial point is taken to be the graphics cursor (in global coordinates).




The command byte tells the terminal whether the line style byte will be sent and which of the four end-point bytes will be sent. This information is bit encoded within the command byte. A bit in the command byte is set if one of the bytes is to follow and is reset if the byte will not follow. The command byte bit structure is as follows:

| Bit Number | Byte to Follow |
|---|---|
| 0 | Low X |
| 1 | High X |
| 2 | Low Y |
| 3 | High Y |
| 4 | Line Style |
| 5 | Reserved |
| 6 | Always Set |
| 7 | Always Reset |

The line style code is encoded in the least significant 3 bits of the Line Style Byte. Bit number 6 of the Line Style Byte is always set, and bit number 7 is always reset. Bits 3 through 5 are ignored, but should be reset. The line styles selected by bits 0, 1, and 2 are as follows:

| Bit Number 2 1 0 | Line Style |
|---|---|
| 0 0 0 | Solid White |
| 0 0 1 | Long Dash |
| 0 1 0 | Short Dash |
| 0 1 1 | Dotted |
| 1 0 0 | Dot-Dashed |
| 1 0 1 | Sparse-Dot |
| 1 1 0 | Solid Black |
| 1 1 1 | Solid Black |

Fast vector mode is exited when the command byte specifies that no bytes will follow.

# SECTION 4
# COMMAND DESCRIPTIONS

## 4.1 INTRODUCTION

The commands described in this section fall into two broad categories. In the first, are all commands that the terminal has in common with the VT100-type terminals. The second category contains commands used to control the special features of the ST3220 terminal.

VT100/VT220 commands control typical alphanumeric terminal features such as cursor positioning, scrolling, and character attributes. The ST3220 terminal supports many of the commands found in these popular terminals.

The ST3220 special commands control features that are not found in a typical alphanumeric terminal. There are commands to draw graphic objects, control the raster (screen image) memory, and control optional features such as "Display List Memory."

All commands are encoded in the ANSI 3.64 standard. The best known example of a device using ANSI 3.64 commands is the VT100 terminal. The standard simply defines a way to pass information and commands to terminal-display devices.

The command descriptions which follow are listed first in a brief summary, then in a second, more detailed description in which the command and its parameters are defined and explained.

## 4.2 DEFINITIONS

ANSI Command Structure examples:

ESC F
ESC <space> F
CSI Pi ; Pk : ... F
DCS ... ST

Where:
1. ESC is ASCII character "escape", code 1BH.
2. CSI is ANSI symbol for code 9BH or the sequence 1BH 5BH.
3. DCS is ANSI symbol for code 90H or the sequence 1BH 50H.
4. ST is ANSI symbol for code 9CH or the sequence 1BH 5CH.
5. Pi, Pk are command parameters and represent ASCII encoded decimal numbers.
6. Semi-colons are used to terminate positive parameters.
7. Colons are used to terminate negative parameters.
8. F is a final character, ASCII codes 40H through 7FH.

***NOTES:***
1. Spaces are used in command descriptions only for illustration, they are not part of the command. If an ASCII space character is part of a command, it will be shown as "<space>."

2. For a detailed description of ANSI commands, see Section 3, Paragraph 3.1.

## 4.3 COMMAND SUMMARY

### 4.3.1 Supported VT100/VT220 Commands:

| | Command | Command Sequence |
|---|---|---|
| SCROLLING | Scroll Up | CSI[1] Pn S |
| | Scroll Down | CSI Pn T |
| | Scroll Margins | CSI Pt ; Pb r |
| CURSOR POSITIONING | Cursor Up | CSI Pn A |
| | Cursor Down | CSI Pn B |
| | Cursor Right | CSI Pn C |
| | Cursor Left | CSI Pn D |
| | Cursor Position | CSI Pl ; Pc H |
| | Cursor Home | CSI H |
| | Horiz. & Vert. Position | CSI Pl ; Pc f |
| | Index | ESC D |
| | Reverse Index | ESC M |
| | Next Line | ESC E |
| CHARACTER ATTRIBUTES | All Attributes Off | CSI m or CSI 0 m |
| | Reverse Video On | CSI 7 m |
| | Reverse Video Off | CSI 27 m |
| | Multiple Attr. On/Off | CSI Ps1 ; Ps2 ; .. m |
| EDITING | Erase Line From Cursor | CSI K or CSI 0 K |
| | Erase Line To Cursor | CSI 1 K |
| | Erase Line | CSI 2 K |
| | Erase Screen From Cursor | CSI J or CSI 0 J |
| | Erase Screen To Cursor | CSI 1 J |
| | Erase Screen | CSI 2 J |
| PRINTING | Auto Print On | CSI ? 5 i |
| | Auto Print Off | CSI ? 4 i |
| | Print Control On | CSI 5 i |
| | Print Control Off | CSI 4 i |
| | Print Cursor Line | CSI ? 1 i |
| | Print Screen | CSI i or CSI 0 i |
| REPORTS | Device Status | CSI 5 n |
| | Printer Status | CSI ? 15 n |
| | Cursor Pos. Report | CSI 6 n |
| | Identify Terminal | ESC Z |
| RESET | Reset To Initial State | ESC c |
| TABS | Horiz. Tab Set | ESC H |
| | Horiz. Tab Clear | CSI g or CSI 0 g |
| | Clear All Tabs | CSI 3 g |
| SET MODE (ANSI) | Lock Keyboard | CSI 2 h |
| | CR & LF (New Line) | CSI 20 h |
| SET MODE (DEC) | ANSI Mode | CSI ? 2 h |
| | Reverse Video (Screen) | CSI ? 5 h |
| | Auto Wrap On | CSI ? 7 h |
| | Auto Repeat On | CSI ? 8 h |
| | Print Formfeed | CSI ? 18 h |
| | Print Extent = Screen | CSI ? 19 h |
| | Text Cursor On | CSI ? 25 h |

[1] ASCII EQUIVALENT: ESC [

| | Command | Command Sequence |
|---|---|---|
| RESET MODE (ANSI) | Unlock Keyboard | CSI 2 l |
| | CR Only | CSI 20 l |
| RESET MODE (DEC) | VT52 Mode | CSI ? 2 l |
| | Normal Video (Screen) | CSI ? 5 l |
| | Auto Wrap Off | CSI ? 7 l |
| | Auto Repeat Off | CSI ? 8 l |
| | No Print Formfeed | CSI ? 18 l |
| | Print Extent = Scroll | CSI ? 19 l |
| | Text Cursor Off | CSI ? 25 l |
| EIGHT BIT C1 CONTROL CODES | Transmit 7-bit Equiv. | ESC <space> F |
| | Transmit 8-bit Control | ESC <space> G |

VT52 ESCAPE SEQUENCES

Note: These commands are only accessible when in VT52 mode. VT52 mode is entered by the "VT52 Mode" command of the DEC reset mode commands. When in VT52 mode, these are the only commands available. To use the other commands, it is necessary to exit VT52 mode by issuing the "ANSI Mode" command (below).

| | |
|---|---|
| Cursor Up | ESC A |
| Cursor Down | ESC B |
| Cursor Right | ESC C |
| Cursor Left | ESC D |
| Enter Graphics | ESC F |
| Exit Graphics | ESC G |
| Cursor Home | ESC H |
| Reverse Line Feed | ESC I |
| Erase To End Of Screen | ESC J |
| Erase To End Of Line | ESC K |
| Direct Cursor Address | ESC Y Line Column |
| Identify Terminal | ESC Z |
| Alternate Keypad | ESC = |
| Numeric Keypad | ESC > |
| ANSI Mode | ESC < |
| Auto-print On | ESC ^ |
| Auto-print Off | ESC _ |
| Print Control On | ESC W |
| Print Control Off | ESC X |
| Print Screen | ESC ] |
| Print Cursor Line | ESC V |

### 4.3.2 DeeCO Extension to VT100/VT220 Commands:

| | | |
|---|---|---|
| SET/RESET MODE (DeeCO) | Screen Saver On | CSI < 1 h |
| | Screen Saver Off | CSI < 1 l |
| | Cursor Block Style | CSI < 2 h |
| | Cursor Underline Style | CSI < 2 l |
| | Print Input Enable | CSI < 4 h |
| | Print Input Disable | CSI < 4 l |
| | Print Input Buffer | CSI < 5 h |
| | Print Input No Buffer | CSI < 5 l |
| | Clipping Enable | CSI < 6 h |
| | Clipping Disable | CSI < 6 l |
| | Graph String Control On | CSI < 7 h |
| | Graph String Control Off | CSI < 7 l |
| | Graph String Wrap | CSI < 8 h |
| | Graph String No-Wrap | CSI < 8 l |
| | Graph Cursor On | CSI < 9 h |
| | Graph Cursor Off | CSI < 9 l |
| | Graph Cursor Box Type | CSI < 10 h |
| | Graph Cursor X Type | CSI < 10 l |

| | Command | Command Sequence |
|---|---|---|
| SET/RESET MODE (DeeCO) (con't.) | XON/XOFF Enable | CSI < 12 h |
| | XON/XOFF Disable | CSI < 12 l |
| | Set-up Key Enable | CSI < 18 h |
| | Set-up Key Disable | CSI < 18 l |
| REPORT | Print Buffer Request | CSI < 1 n |
| | Print Buffer Response | DCS[2] b <a..a> ST[3] |
| | Auxiliary Memory Checksum Request | CSI < 2 n |
| | Auxiliary Memory Checksum Response | CSI < 2; Pr; Pc n |
| | ST3220 System Status Request | CSI < 3 n |
| | ST3220 System Status Response | CSI < 3; Pv; Pr; Pm; Pi; Ps; Pa n |
| PRINTING | Print Port Configure | CSI < Pb; Pd; Ps; Pp c |
| ALPHA CONTROL | Alpha Character Size | CSI < Ps s |
| | Character Set Select | CSI < Ps S |
| | Alpha Write Mode | CSI < Ps M |
| CHARACTER ATTRIBUTES | Italic Attribute On: | CSI < 9 m |
| | Italic Attribute Off: | CSI < 29 m |
| GRAPH WINDOW COMMAND CONTROL | Define Window Area | CSI < Pyl; Pxl; Pyu; Pxr w |
| | Define Window Origin | CSI < Pyo; Pxo o |
| | Define Window Type | CSI < Pt W |
| | Erase Graphic Window | CSI < e (and:DCS O G ST or DCS G ST) |
| VIEWPORT CONTROL | Define Viewport | CSI < Pv; Ps v |
| | Locate Viewport | CSI < Pv; Pt; Py; Px p |
| DATA/DISPLAY LISTS | Open Display/Data List | CSI < Pid; Pt; Pa d |
| | Close List | Control-\ (1CH) |
| | Execute Display List | CSI < Pid; Px; Py x |
| | Append Data List | DCS Pid a <h..h> ST |
| | Read Data List | CSI < Pid r |
| | List Report | DCS Pid; Ps; Pt r <h..h> ST |
| | Clear All Lists | CSI < C |
| AUXILIARY MEMORY | Auxiliary Memory Type | CSI < Pt m |
| | Compute/Store Checksum | CSI < a |
| CHARACTER DEFINITION | Define Alternate Character | DCS 33 G Pc; P10; P11; ... P17 ST |
| OTHER | Screen Saver Timeout | CSI < Pn t |
| GRAPHIC OBJECT | Clear Graphic Window | DCS 0 ST or DCS G ST |
| | Set Graphic Cursor | DCS 1 G Py; Px ST |
| | Vector Draw, Absolute | DCS 2 G Py1; Px1; Py2; Px2 ST |
| | Vector Draw, Relative | DCS 3 G Py2; Px2 ST |
| | Circle Draw, Absolute | DCS 4 G Pr; Pyc; Pxc ST |
| | Circle Draw, Relative | DCS 5 G Pr ST |
| | Arc Draw, Absolute | DCS 6 G Pa0; Pa1; Pr; Pyc; Pxc ST |
| | Arc Draw, Relative | DCS 7 G Pa0; Pa1; Pr ST |
| | Block Fill Draw, Absolute | DCS 8 G Py1; Px1; Py2; Px2 ST |
| | Block Fill Draw, Relative | DCS 9 G Py2; Px2 ST |
| | Concave Polygon Fill, Absolute | DCS 10 G Py; Px ST |
| | Concave Polygon Fill, Relative | DCS 11 G ST |
| | Convex Polygon Fill, Absolute | DCS 12 G Py; Px ST |
| | Convex Polygon Fill, Relative | DCS 13 G ST |

[2] ASCII EQUIVALENT: ESC P
[3] ASCII EQUIVALENT: ESC \

| | Command | Command Sequence |
|---|---|---|
| GRAPHIC OBJECT (con't.) | Graphic String Draw, Absolute | DCS 14 G Py; Px A <....> ST |
| | Graphic String Draw, Relative | DCS 15 G A <....> ST |
| GRAPHIC ATTRIBUTES | Character Attribute Select | DCS 16 G Ps; Pr; Pi; Pu; Pc ST |
| | String Attribute Select | DCS 17 G Pd; Pp ST |
| | Line Style Attribute Select | DCS 18 G Ps ST |
| | Fill Style Attribute Select | DCS 19 G Ps ST |
| | Graph Write Mode Select | DCS 20 G Ps ST |
| RASTER OPERATIONS | Raster Memory Write, Absolute | DCS 21 G Py; Px H <h..> ST |
| | Raster Memory Write, Relative | DCS 22 G H <h..> ST |
| | Raster Memory Read Request, Absolute | DCS 23 G Py; Px; Pc ST |
| | Raster Memory Read Request, Relative | DCS 24 G Pc ST |
| | Raster Memory Read Response | DCS 25 G <h..> ST |
| AREA FLASH | Flash Area, Define | DCS 30 G Pi; Pl; Pyl; Pxl; Pyu; Pxr ST |
| | Flash Area, Start | DCS 31 G Pi ST |
| | Flash Area, Stop | DCS 32 G Pi ST |
| FAST VECTOR | Enter Fast Vector Mode | DCS 34 G ST <vector data ..> |

## 4.4 DETAILED COMMAND DESCRIPTIONS:

### SCROLL

SCROLL UP: CSI Pn S
Function: Cause the scroll region to scroll up.
Pn: Number of lines to scroll.

SCROLL DOWN: CSI Pn T
Function: Cause the scroll region to scroll down.
Pn: Number of lines to scroll.

SCROLL MARGINS: CSI Pt; Pb r
Function: Select the top and bottom of the scrolling region (that area of the screen that can scroll as new characters are received). The scroll region must consist of at least 2 lines. Lines are counted from "1". If no value is specified, or after power-up or reset, the default values are the top and bottom of the screen.
Pt: Line number of top of scroll region.
Pb: Line number of bottom of scroll region.

### CURSOR POSITIONING

CURSOR UP: CSI Pn A
Function: Move the cursor up. The cursor will stop at the top of the scroll region.
Pn: Number of lines to move up.

CURSOR DOWN: CSI Pn B
Function: Move the cursor down. The cursor will stop at the bottom of the scroll region.
Pn: Number of lines to move down.

CURSOR RIGHT: CSI Pn C
Function: Move the cursor right. The cursor will stop at the right margin.
Pn: Number of character positions to move.

CURSOR LEFT: CSI Pn D
Function: Move the cursor left. The cursor will stop at the left margin.
Pn: Number of character positions to move.

CURSOR POSITION: CSI Pl; Pc H
Function: Move the cursor to the specified character line and column. Rows are numbered 1 through 24, columns from 1 to 80. Row or column 0 is interpreted as row or column 1. Default for a missing parameter is 1. If the row or column specified is greater than the actual number, then the largest number possible is used.
Pl: Line number.
Pc: Column number.

CURSOR HOME: CSI H
Function: Returns the text cursor to line 1, column 1. The contents of the screen are not changed.

HORIZ. & VERT. POSITION: CSI Pl; Pc f
Function: Same as Cursor Position. Cursor Position is preferred.
Pl: Line number.
Pc: Column number.

INDEX: ESC D
Function: Move the cursor to the line below, but keep it in the same column. If the cursor is already at the bottom of the scroll region, the screen scrolls up a line. Index is also an 8-bit control character, 84H.

REVERSE INDEX: ESC M
Function: Move the cursor to the line above, but keep it in the same column. If the cursor is already at the top of the scroll region, the screen scrolls down a line. Reverse Index is also an 8-bit control character, 8DH.

NEXT LINE: ESC E
Function: Move the cursor to the start of the next line. If the cursor is already at the bottom of the scroll region, the screen scrolls up a line. Next Line is also an 8-bit control character, 85H.

## CHARACTER ATTRIBUTES

ALL ATTRIBUTES OFF: CSI m or CSI 0 m
REVERSE VIDEO ON: CSI 7 m
REVERSE VIDEO OFF: CSI 27 m
ITALIC ON: CSI 9m
ITALIC OFF: CSI 29m
Function: Select or deselect the specified attribute for all new characters received. If the characters are subsequently moved, the attributes move with them.

MULTIPLE ATTRIB. ON/OFF: CSI Ps1; Ps2; .. m
Function: Multiple attribute selection is performed one at a time in the sequence specified.
Ps[i]: 7,9,27, or 29.

## EDITING

ERASE LINE FROM CURSOR: CSI K or CSI 0 K
Function: Erase from the cursor (inclusive) to the end of the line. No line attributes are affected.

ERASE LINE TO CURSOR: CSI 1 K
Function: Erase from the beginning of the line to the cursor, inclusive. No line attributes are affected.

ERASE LINE: CSI 2 K
Function: Erase the entire line.

ERASE SCRN FROM CURSOR: CSI J or CSI 0 J
Function: Erase from the cursor (inclusive) to the end of the screen.

ERASE SCRN TO CURSOR: CSI 1 J
Function: Erase from the beginning of the screen to the cursor, inclusive. Fully erased lines are made single-height, single-width.

ERASE SCREEN: CSI 2 J
Function: Erase the entire screen, changing all lines to single height single width. The cursor maintains its original position.

## PRINTING

**Note:** Before using any print commands, check printer status (see "Reports"). After the last printable character in a line, a carriage return & linefeed (CR,LF) or vertical tab (VT) or form feed (FF) is sent.

AUTO PRINT ON: CSI ? 5 i
Function: Auto-print mode causes a line to be printed when the cursor advances to the next line due to a linefeed, formfeed, vertical tab, or auto-wrap (CR,LF). The printed line ends with a carriage return and the LF, FF, or VT.

AUTO PRINT OFF: CSI ? 4 i
Function: Turn off auto-print mode.

PRINT CURSOR LINE: CSI ? 1 i
Function: Print the line containing the cursor, without moving the cursor.

PRINT CONTROL ON: CSI 5 i
Function: Print control mode causes host input to pass through directly to the printer without being displayed on screen. Exceptions: NUL, XON, and XOFF are not sent to the printer. Keyboard input continues to be sent to the host. Print control mode can be invoked while in auto print mode, but not vice versa.

PRINT CONTROL OFF: CSI 4 i
Function: Turn off print control mode.

PRINT CURSOR LINE: CSI ? 1 i
Function: Prints the display line containing the cursor. The cursor position does not change. The print cursor line sequence is complete when the line prints.

PRINT SCREEN: CSI i or CSI 0 i
Function: Print either the entire screen, or just the scroll region, as determined by the state of the print extent mode. There is either no print terminator character, or a form feed, depending on the state of the printer form feed mode.

## REPORTS

DEVICE STATUS: CSI 5 n
Function: Host requests operating status of the terminal and/or printer. (If the terminal is in Print Control mode, the request goes to the printer, which may be unable to respond.)
Responses: No malfunction: CSI 0 n
Malfunction: CSI 3 n

PRINTER STATUS: CSI ? 15 n
Function: To determine printer status (required before sending any print or print mode commands to the terminal).
Responses: Printer ready (DTR): CSI ? 10 n
Printer not ready (no DTR): CSI ? 11 n
No printer (no DTR since power-up or reset): CSI ? 13 n

CURSOR POS. REPORT: CSI 6 n
Function: Request cursor vertical and horizontal position.
Response: CSI Pv; Ph R
Pv: Vertical position (row, counting from 1)
Ph: Horizontal position (column, counting from 1)

## RESET

RESET TO INITIAL STATE: ESC c
Function: Cause a hard terminal reset.

## TABS

HORIZ TAB SET: ESC H
Function: Set a tab stop at the current cursor position.

HORIZ TAB CLEAR: CSI g or CSI 0 g
Function: Clear a tab stop at the current cursor position.

CLEAR ALL TABS: CSI 3 g
Function: Clear all tab stops.

## SET MODE (ANSI)

LOCK KEYBOARD: CSI 2 h
Function: Cause keyboard input to be ignored.

NEWLINE: CSI 20 h
Function: Append a linefeed to each carriage return (or enter key, when the numeric keypad is in numeric mode). Append a carriage return to each linefeed (LF).

## SET MODE (DEC)

ANSI MODE: (none)
**Note:** When in VT52 mode, the terminal cannot respond to ANSI sequences, so there is no ANSI set command for this mode. To get back to ANSI mode, use the VT52 command "ESC <".

REVERSE VIDEO (SCREEN): CSI ? 5 h
Function: Reverse the video image of the entire screen.

AUTO WRAP ON: CSI ? 7 h
Function: If the cursor is at the right margin when a new character is received, precede the character with a carriage return and linefeed. If the cursor is at the bottom of the scroll region, cause a scroll-up.

AUTO REPEAT ON: CSI ? 8 h
Function: Cause a key depressed more than 0.5 seconds to be repeated, except for these keys: HOLD SCREEN, PRINT SCREEN, BREAK, RETURN, LOCK, SHIFT and CONTROL.

PRINT FORMFEED: CSI ? 18 h
Function: After a screen print, cause a formfeed (FF) to be sent.

PRINT EXTENT=SCREEN: CSI ? 19 h
Function: Causes print-screen function to print the entire screen rather than just scroll region.

TEXT CURSOR ON: CSI ? 25 h
Function: Turn on the cursor.

## RESET MODE (ANSI)

UNLOCK KEYBOARD: CSI 2 l
Function: Unlock the keyboard.

CR ONLY: CSI 20 l
Function: No linefeeds appended to carriage returns. No carriage returns appended to linefeed (LF).

## RESET MODE (DEC)

VT52 MODE: CSI ? 2 l
Function: Respond like a VT52.

NORMAL VIDEO (SCREEN): CSI ? 5 l
Function: Normal (not reverse) video.

AUTO WRAP OFF: CSI ? 7 l
Function: Off.

AUTO REPEAT OFF: CSI ? 8 l
Function: Off.

NO PRINT FORMFEED: CSI ? 18 l
Function: No formfeed after a screen print.

PRINT EXTENT=SCROLL: CSI ? 19 l
Function: Causes print-screen function to print only the scroll region.

TEXT CURSOR OFF: CSI ? 25 l
Function: Cursor off.

## EIGHT-BIT C1 CONTROL CODES

TRANSMIT 7-BIT EQUIV: ESC <space> F
<space>: ASCII space character

TRANSMIT 8-BIT CNTRL: ESC <space> G
<space>: ASCII space character

## VT52 ESCAPE SEQUENCES

Note: These commands are only accessible when in VT52 mode. VT52 mode is entered by the "VT52 Mode" command of the DEC reset mode commands. When in VT52 mode, these are the only commands available. To use the other commands, it is necessary to exit VT52 mode by issuing the "ANSI Mode" command (below).

CURSOR UP: ESC A
Function: Move the cursor up one character row.

CURSOR DOWN: ESC B
Function: Move the cursor down one character row.

CURSOR RIGHT: ESC C
Function: Move the cursor right one character column.

CURSOR LEFT: ESC D
Function: Move the cursor left one character column.

CURSOR HOME: ESC H
Function: Move the cursor to the "home" position, row 1, column 1.

ENTER GRAPHICS: ESC F
Function: Select the graphic character set. This set is the user-definable alternate character set.

EXIT GRAPHICS: ESC G
Function: Select the standard character set.

REVERSE LINEFEED: ESC I
Function: Move the cursor up one character row. If the cursor is on row 1, then scroll the display down one line inserting a new blank line for row 1.

ERASE TO END OF SCRN: ESC J
Function: Erase from the cursor to the end of the screen.

ERASE TO END OF LINE: ESC K
Function: Erase from the cursor to the end of the current line

DIRECT CURSOR ADDRESS: ESC Y Line Column
Function: Move the cursor to a specific line and column. The line and column numbers are specified by single ASCII characters. The characters are found by adding the row number (1 through 24) or the column number (1 through 80) to the value 31. The resulting number represents a displayable ASCII character.

IDENTIFY TERMINAL: ESC Z
Function: Terminal type request. Display responds with VT52 type, the sequence "ESC / Z" (spaces not sent).

ANSI MODE: ESC <
Function: Return to ANSI mode of operation (exit VT52 mode).

AUTO PRINT ON: ESC ^
Function: Turn auto-print on. When auto-print is on, a line is automatically printed when a linefeed is issued, moving the cursor off the line.

AUTO PRINT OFF: ESC _
Function: Turn auto-print off.

PRINT CONTROL ON: ESC W
Function: Turn print-control on. When print-control is on, all input is directed to the printer until the Print Control Off command is received.

PRINT CONTROL OFF: ESC X
Function: Turn print-control off.

PRINT SCREEN: ESC ]
Function: Print the entire screen.

PRINT CURSOR LINE: ESC V
Function: Print the current line.

## SET/RESET MODE (DeeCO)

SCREEN SAVER ON: CSI < 1 h
SCREEN SAVER OFF: CSI < 1 l
Function: Enable the screen saver time-out feature. This feature automatically blanks the screen after a specified number of minutes of inactivity (no host, keyboard, or touch input). Screen saver off disables the screen saver time-out feature.

CURSOR BLOCK STYLE: CSI < 2 h
Function: Selects the "box" style of blinking alphanumeric cursor.

CURSOR UNDERLINE STYLE: CSI < 2 l
Function: Selects the "underline" style of blinking alphanumeric cursor.

PRINT INPUT ENABLE: CSI < 4 h
PRINT INPUT DISABLE: CSI < 4 l
Function: Enable (h) or Disable (l) input recognition from the print device. If enabled, all input from the print device is passed on to the host. The data will be passed on immediately or may be buffered until requested by the host, see PRINT INPUT BUFFER MODE below.

PRINT INPUT BUFFER: CSI < 5 h
PRINT INPUT NO BUFFER: CSI < 5 l
Function: Enable (h) or Disable (l) print device input buffering. If enabled, and if print input is enabled, then all input from the printer is buffered by the terminal and may be requested by the host. If buffering is disabled, and print input is enabled, then all input from the print device is passed directly on to the host.

CLIPPING ENABLE: CSI < 6 h
CLIPPING DISABLE: CSI < 6 l
Function: Enables (h) clipping of graphic objects at the graphic window limits. Disable (l) command is the opposite.

GRAPH STRING CONTROL ON: CSI < 7 h
GRAPH STRING CONTROL OFF: CSI < 7 l
Function: When on, all ASCII control characters are interpreted as appropriate cursor control functions. When off, ASCII control codes are treated as printable characters. Pertains only to the graphic string draw.

GRAPH STRING WRAP: CSI < 8 h
GRAPH STRING NO-WRAP: CSI < 8 l
Function: When on, any attempt to draw a character outside of the graphic window will cause the graphic cursor to wrap to the opposite border. When off, characters will be clipped beyond the window. Pertains only to the graphic string draw.

GRAPH CURSOR ON: CSI < 9 h
GRAPH CURSOR OFF: CSI < 9 l
Function: Enable or disable the display of the graphics cursor symbol.

GRAPH CURSOR BOX TYPE: CSI < 10 h
GRAPH CURSOR CROSS (X) TYPE: CSI < 10 l
Function: Selects the graphic cursor symbol. The box symbol is similar to the alpha box cursor. Its size is controlled by the graphics character size. The cross type is always a 7 X 7 cross.

XON/XOFF Enabled (Default): CSI < 12 h
XON/XOFF Disabled: CSI < 12 l
Function: When enabled the ST3220 will send software handshaking characters to the host. If disabled, no software handshaking characters will be sent.

SET-UP KEY Enabled (Default): CSI < 18 h
SET-UP KEY Disabled: CSI < 18 l
Function: To prevent an operator from changing set-up states, it is possible to disable the set-up key on the on-screen and external keyboard.

## REPORT

PRINT BUFFER REQUEST: CSI < 1 n
Function: Host request for any buffered input from the print device.

PRINT BUFFER RESPONSE: DCS b <a..a> ST
Function: Terminal report of buffered input from the print device.
<a..a>: Input from the print device.

AUXILIARY MEMORY CHECKSUM REQUEST: CSI < 2 n

Function: Request for report on the auxiliary memory checksum test.

AUXILIARY MEMORY CHECKSUM RESPONSE: CSI < 2; Pr, Ps, Pc n

Function: Response to report request of the auxiliary memory checksum test. The result of the checksum test is reported with two numbers, the checksum and the result of the checksum match. The auxiliary memory checksum is stored in the first byte of the auxiliary memory device. The checksum is the module 256 (8-bit) sum of all bytes in the memory except the first byte. The checksum reported is the calculated value, the result reported is 1 if that value is stored in the first memory location and 0 if it is not.
Pr: Result (0 or 1) of the checksum test.
Ps: Size of auxiliary memory; 2=8Kx16, 8=32Kx16
Pc: Calculated checksum.

ST3220 SYSTEM STATUS REQUEST: CSI < 3 n

Function: Request for the ST3220 system report.

ST3220 SYSTEM STATUS RESPONSE: CSI < 4; Pv; Pr; Pm; Pi; Ps; Pa n

Function: Response to the request for the ST3220 system report. The response includes the firmware version and revision numbers, and the status of optional ST3220 features such as I-R touch, auxiliary memory, 32K system RAM, and the secondary serial port. Each option is reported as a number ending with 0 or 1.
Pv: Firmware version number.
Pr: Firmware revision number.
Pm: System RAM option- 10 = 8K system RAM, 11 = 32K system RAM.
Pi: I-R touch option- 20 = no I-R touch detected, 21 = I-R touch detected.
Ps: Secondary serial port option- 30 = no second serial, 31 = second serial detected.
Pa: Auxiliary memory option- 40 = no valid auxiliary memory device detected, 41 = valid auxiliary memory device (initialization sequence) detected.

## PRINTING

PRINTER PORT CONFIGURE: CSI < Pb; Pd; Ps; Pp c

Function: Configure the printer port serial IO.
Pb: Baud rate selection (0=19200,1=9600,2=4800,3=2400,4=1200, 5=600,6=300,7=110).
Pd: Number of data bits (0=8 bits,1=7 bits,2=6 bits,3=5 bits).
Ps: Number of stop bits (0=1 bit,1=1.5 bits,2=2 bits).
Pp: Parity selection (0=none, 1=odd, 2=even).

## ALPHA CONTROL

ALPHA CHARACTER SIZE: CSI < Ps s

Function: Select the character size used for terminal characters. There are four sizes available, the basic size and 3 additional sizes. The basic size corresponds to a 8x16 character cell. The additional sizes multiply that basic size by 2, 4, and 8.
Ps: Character size selection (1, 2, 3, or 4).

CHARACTER SET SELECT: CSI < Ps S

Function: Select the standard or alternate character set for active use. The alternate character set is user-defined and requires the optional auxiliary memory.
Ps: Character set selection, 0=standard, 1=alternate.

## GRAPH WINDOW COMMAND CONTROL

ALPHA WRITE MODE: CSI < Ps M

Function: Selects the raster write mode for terminal alphanumerics. The write mode may be JAM, COMPLEMENT, SET, or CLEAR.
Pm: Write mode (0=JAM, 1=COMPLEMENT, 2=CLEAR, 3=SET).

DEFINE WINDOW AREA: CSI< Pyl; Pxl; Pyu; Pxr w

Function: Define the rectangular bounds of a graphic window. The bounds are given as relative offsets from the window origin. The rectangle defined by these offsets are used as the clipping rectangle when clipping is enabled.
Pyl: Y coordinate, lower left corner (relative to window origin).
Pxl: X coordinate, lower left corner (relative to window origin).
Pyu: Y coordinate, upper right corner (relative to window origin).
Pxr: X coordinate, upper right corner (relative to window origin).

DEFINE WINDOW ORIGIN: CSI < Pyo; Pxo o

Function: Define the window origin point. The origin is given in global coordinates. This command will cause any viewport tied to the window origin to be updated.
Pyo: Y coordinate, window origin (relative to global origin).
Pxo: X coordinate, window origin (relative to global origin).

DEFINE WINDOW TYPE: CSI < Pt W

Function: Select the window type, graphic or Alpha. A graphic window is defined by the graphic window origin and graphic window area. An alpha window is defined to be the alpha page. It uses the lower left corner of the alpha page as its origin and the page extent as its area. An alpha window allows objects to be drawn consistent with alpha text on the alpha page. The window area is always set to a full page size and the window origin is set to the global origin when the window type changes.
Pt: Type 0=Graphic, 1=Alpha

ERASE GRAPHIC WINDOW: CSI < e (and: DCS O G ST or DCS G ST)

Function: Erase the area encompassed by the graphic window.

## VIEWPORT CONTROL

DEFINE VIEWPORT: CSI < Pv; Ps v

Function: Define the physical aspects of a viewport. This command causes the viewport to be updated on the screen if the size is non-zero.
Pv: viewport number.
Ps: size (height) of the viewport.

LOCATE VIEWPORT: CSI < Pv; Pt; Py; Px p

Function: Set the viewport memory origin in raster memory. This command causes the viewport to be updated on the screen if the size is non-zero. When defining a new viewport, this command should be used before the DEFINE VIEWPORT command to insure that the proper portion of raster memory is always displayed. The Y and X offsets specified are added to the start position implied by the viewport type and are optional.
Pv: viewport number, 0 or 1.
Pt: viewport type; Graph Absolute (0), Graph Relative (1), Alpha Absolute (2), Alpha Relative (3).
Py: Y coordinate offset.
Px: X coordinate offset.

## DATA/DISPLAY LISTS

OPEN DISPLAY/DATA LIST: CSI < Pid; Pa Pt; d

Function: Open a list for write access. Only one list may be open for writing at one time.
Pid: D number of the list. Any previous list with this number is deleted. This may fragment list memory if the body directly follows the header.
Pt: List type; 0=display list, 1=data list. Defaults to 0 if this parameter is missing.
Pa: List address. This parameter specifies the start of the list body in list memory. If missing, the list body will start immediately after the list header.

CLOSE LIST: ASCII Control Code 1CH

Function: Close the current write-open list.

EXECUTE DISPLAY LIST: CSI < Pid; Py; Px x

Function: Open a display list for reading, and execute the commands in the list as if they were host input.
Pid: ID number of the list.
Py: Y offset from the origin of all graphic commands.
Px: X offset from the origin of all graphic commands.

APPEND DATA LIST: DCS Pid a <h...h> ST

Function: Open a data list and append to the end of the list. The data list must have been created by a previous "open list" command, then properly closed.
Pid: ID of the list. <h..h>: Data, in hex-ASCII, to append to the list.

READ DATA/DISPLAY LIST: CSI < Pid r

Function: Open a list for reading and report the data to the host.
Pid: ID of the list.

LIST REPORT: DCS Pid; Ps; Pt r <h..h> ST

Function: Report format requested by the READ DATA/DISPLAY LIST command.
Pid: ID of the list.
Ps: Size (in bytes) of the list. <h..h>: Data, in hex-ASCII, of the list.
Pt: List type (0 for display, 1 for data)

CLEAR ALL LISTS: CSI < C

Function: Clears the auxiliary memory list structure completely. This command must be used to initialize a new EEPROM.

COMPUTE/STORE CHECKSUM: CSI < a or CSI < 1a

Function: Computes and stores auxiliary memory checksum at first address. CSI < 1a computes and stores checksum and verifies initialization.

## AUXILIARY MEMORY

AUXILIARY MEMORY TYPE: CSI < Pt m

Function: Auxiliary memory may be EEPROM or RAM. The ST3220 terminal always assumes that it is EEPROM. This command may be used to tell the terminal that the memory type is RAM or EEPROM. When the type is EEPROM, all writes to the auxiliary memory force the terminal to wait for approximately 10ms before accessing the memory again. If the type is RAM, the terminal does not wait between accesses.
Pt: Auxiliary memory type (0=EEPROM, 1=RAM).

## CHARACTER DEFINITION

DEFINE ALTERNATE CHARACTER: DCS 33 G Pc; P0; P1; P2; .. Pn ST

Function: Define the bit pattern of an alternate character. The character is defined in an 8xn pixel array, where n is the character cell height. The character code defines where in the alternate character memory the character pattern will be stored. The 8xn pattern maps to the normal character image with the first byte sent corresponding to the top of the character, and the lsb's of the bytes corresponding to the left edge of the character.
Pc: Character code of the character being defined.
P0-Pn: 8 bits of character image.

## OTHER

SCREEN SAVER TIMEOUT: CSI < Pn t

Function: Specify the time, in minutes, of the screen saver time-out feature. This is the time of allowable inactivity before the screen is automatically blanked by the screen saver feature. The minimum time allowable is 1 minute, the maximum is 20 minutes.

## GRAPHIC OBJECT

Note: All coordinates given are relative coordinates, that is relative to the window origin.

CLEAR GRAPHIC WINDOW: DCS 0 G ST or DCS G ST

Function: Erase the area encompassed by the graphic window.

SET GRAPHIC CURSOR: DCS 1 G Py; Px ST

Function: Set the graphics cursor to a specific location.
Py: Y coordinate of cursor.
Px: X coordinate of cursor.

VECTOR DRAW, ABSOLUTE: DCS 2 G Py1; Px1; Py2; Px2 ST

Function: Draw a vector from (Y1,X1) to (Y2,X2). The new graphics cursor becomes (X2,Y2). The vector appearance is controlled by the line style attribute.
Py1: Y of start position of vector.
Px1: X of start position of vector.
Py2: Y of end position of vector.
Px2: X of end position of vector.

VECTOR DRAW, RELATIVE: DCS 3 G Py2; Px2 ST

Function: Draw a vector from the current graphics cursor to the graphics cursor plus (Y2,X2). The new graphics cursor becomes the graphics cursor plus (Y2,X2). The vector appearance is controlled by the line style attribute.
Py2: Y of end position of vector.
Px2: X of end position of vector.

CIRCLE DRAW, ABSOLUTE: DCS 4 G Pr; Pyc; Pxc ST

Function: Draw a circle, centered at (Yc,Xc) with radius Pr. The graphics cursor becomes the circle center upon completion. The border is controlled by the line style attribute.
Pr: Radius of circle.
Pyc: Y of circle center position.
Pxc: X of circle center position.

CIRCLE DRAW, RELATIVE: DCS 5 G Pr ST

Function: Draw a circle, centered at the current graphics cursor with radius Pr. The graphics cursor remains unchanged. The border is controlled by the line style attribute.
Pr: Radius of circle.

ARC DRAW, ABSOLUTE: DCS 6 G Pa1; Pa2; Pr; Pyc; Pxc ST

Function: Draw a circular arc, centered at (Yc,Xc) with radius Pr. The arc start and stop are specified as angles from the horizontal (0 degrees at the right). The graphics cursor becomes the circle center upon completion. The border is controlled by the line style attribute.

Pa1: Angle, in degrees, of arc start.
Pa2: Angle, in degrees, of arc stop.
Pr: Radius of circle.
Pyc: Y of circle center position.
Pxc: X of circle center position.

ARC DRAW, RELATIVE: DCS 7 G Pa1; Pa2; Pr ST

Function: Draw a circular arc, centered at the current graphics cursor with radius Pr. The arc start and stop are specified as angles from the horizontal (0 degrees at right). The arc is drawn counterclockwise. The graphics cursor remains unchanged. The border is controlled by the line style attribute.

Pa1: Angle, in degrees, of arc start.
Pa2: Angle, in degrees, of arc stop.
Pr: Radius of circle.

BLOCK FILL DRAW, ABSOLUTE: DCS 8 G Pyl; Pxl; Pyu; Pxr ST

Function: Fill a rectangular area. The fill pattern is controlled by the fill style attribute. The rectangular region is defined by specifying the lower left and upper right corner positions. The graphics cursor becomes the upper right corner position upon completion.

Pyl: Y coordinate, lower left corner.
Pxl: X coordinate, lower left corner.
Pyu: Y coordinate, upper right corner.
Pxr: X coordinate, upper right corner.

BLOCK FILL DRAW, RELATIVE: DCS 9G Pyu; Pxr ST

Function: Fill a rectangular area. The fill pattern is controlled by the fill style attribute. The rectangular region is defined by current graphics cursor (the lower left corner) and the graphics cursor plus (Yu,Yr) . The graphics cursor becomes the upper right corner position upon completion.

Pyu: Y coordinate, upper right corner.
Pxr: X coordinate, upper right corner.

CONCAVE POLYGON FILL, ABSOLUTE: DCS 10 G Pys; Pxs ST

Function: Fill a concave polygonal area. The fill pattern is always a solid color, white or black. The region is defined by a solid border surrounding the specified seed position. If the pixel at the seed position is white, then the border is assumed to be black. If the seed pixel is black, then the border is assumed to be white. The region is filled with the border color. The graphics cursor becomes the seed position upon completion.

Pys: Y coordinate, seed position.
Pxs: X coordinate, seed position.

CONCAVE POLYGON FILL, RELATIVE: DCS 11 G ST

Function: Fill a concave polygonal area. The fill pattern is always a solid color, white or black. The region is defined by a solid border surrounding the specified seed position. If the pixel at the seed position is white, then the border is assumed to be black. If the seed pixel is black, then the border is assumed to be white. The region is filled with the border color. The graphics cursor is used as the seed position and remains unchanged upon completion.

CONVEX POLYGON FILL, ABSOLUTE: DCS 12 G Pys; Pxs ST

Function: Fill a convex polygonal area. The fill pattern is controlled by the fill style attribute. The region is defined by a solid border surrounding the specified seed position. If the pixel at the seed position is white, then the border is assumed to be black. If the seed pixel is black, then the border is assumed to be white. The region is filled with the fill style attribute pattern. The graphics cursor becomes the seed position upon completion.

Pys: Y coordinate, seed position.
Pxs: X coordinate, seed position.

CONVEX POLYGON FILL, RELATIVE: DCS 13 G ST

Function: Fill a convex polygonal area. The fill pattern is controlled by the fill style attribute. The region is defined by a solid border surrounding the specified seed position. If the pixel at the seed position is white, then the border is assumed to be black. If the seed pixel is black, then the border is assumed to be white. The region is filled with the fill style attribute pattern. The graphics cursor is used as the seed position and remains unchanged upon completion.

GRAPHIC STRING DRAW, ABSOLUTE: DCS 14 G Py; Px A <a...a> ST

Function: Draw a text string. The current string attributes control the character orientation and string path. The current character attributes control the character size, set, and font. The graphics cursor becomes the last character position + 1 upon completion of the command.

Py: Y coordinate, string start position.

Px: X coordinate, string start position.

<a..a>: ASCII codes of characters. Control codes are assumed to be displayable except for Escape (code 1BH) and String Terminator (code 9CH) which terminate the command.

GRAPHIC STRING DRAW, RELATIVE: DCS 15 G A <a...a> ST

Function: Draw a text string starting at the current graphics cursor position. The current string attributes control the character orientation and string path. The current character attributes control the character size, set, and font. The graphics cursor becomes the last character position + 1 upon completion of the command. <a..a>: ASCII codes of characters. Control codes are assumed to be displayable except for Escape (code 1BH) and String Terminator (code 9CH) which terminates the command.

## GRAPHIC ATTRIBUTES

CHARACTER ATTRIBUTE SELECT: DCS 16 G Ps; Pr; Pi; Pu; Pc ST

Function: Select the character attributes. These are 1) Size: 1 to 16 for pixel replication character zoom factor; 2) Video format, normal or reverse; 3) Font, normal or slanted; 4) Underline, off or on; and 4) Character set, standard or alternate. The video format, underline, font, and character set are stored on a character by character basis. Size is not stored as character attribute. The character set selected is either the current terminal (alpha) or it is the alternate (user defined) set. These two specifications can yield the same set. Omitted parameters are assumed to be zero.

Ps: Size: 0-15 for 1-16 zoom.

Pr: Reverse Video Attribute: 0 for normal, 1 for reverse.

Pi: Italic Font Attribute: 0 for normal, 1 for slanted (italic).

Pu: Underline Attribute: 0 for no underline, 1 for underline.

Pc: Character Set: 0 for alpha set, 1 for user-defined.

STRING ATTRIBUTE SELECT: DCS 17 G Pd; Pp ST

Function: Select the string attributes. These are 1) Direction: 45 degree rotation of character string; 2) Path: left-to-right, right-to-left, top-to-bottom, or bottom-to-top.

Pd: Direction: 0-7 for multiple of 45 degrees, counterclockwise, from right horizontal. This orients the characters and the entire string.

Pp: Path: 0-3 for left-to-right, bottom-to-top, right-to-left or top-to-bottom.

LINE STYLE ATTRIBUTE, SELECT: DCS 18 G Ps ST

Function: Select the line style for vectors and object borders.

Ps: Style: 0=solid white, 1=long-dash, 2=short-dash, 3=dot, 4=dot-dash, 5=sparse-dot, 6=solid black.

FILL STYLE ATTRIBUTE, SELECT: DCS 19 G Ps ST

Function: Select the fill style for block and polygonal fills.

Ps: Style: 0=solid black, 1=solid white, 2=vertical hatch, 3=horizontal hatch, 4=slant left hatch, 5=slant right hatch, 6=sparse dot.

GRAPH WRITE MODE SELECT: DCS 20 G Pm ST

Function: Select the raster write mode. The write mode refers to how new raster data (objects being drawn) and existing raster data are combined. The simplest mode is "replace" where new data completely replaces the old. There are three other methods, each described by a Boolean logical expression: 1) complement: RASTER = NEW XOR OLD, 2) clear, and 3) set: RASTER = NEW OR OLD:

Pm: Write mode: 0-3 for replace, complement, clear, set

## RASTER OPERATIONS

RASTER MEMORY WRITE, ABSOLUTE: DCS 21 G Py; Px H <h..h> ST

Function: Write directly to the raster memory. The data start is defined by specifying the start position. The data to be written is sent in hex-ASCII. The start position determines a raster memory address. The first datum is written to this location and the rest of the data is written into successive memory locations in 16-bit units. Note that the start position may map to a non-byte boundary, however the terminal will force this position to be on a 16-bit boundary by essentially using a starting Y position which is the greatest multiple of 16 possible given the user-specified starting Y.

Py: Start Y (actual start Y used is Py - (Py mod 8).

Px: Start X.<h...h>: Data in hex-ASCII

RASTER MEMORY WRITE , RELATIVE: DCS 22 G H <h..h> ST

Function: Write directly to the raster memory. The data start is defined by current graphics cursor position. The data to be written is sent in hex-ASCII. The start position determines a raster memory address. The first datum is written to this location and the rest of the data is written into successive memory locations in 16-bit units. Note that the start position may map to a non-byte boundary, however the terminal will force this position to be on a 16-bit boundary by essentially using a starting Y position which is the greatest multiple of 16 possible given the graphics cursor Y position.

<h..h>: Data in hex-ASCII.

RASTER MEMORY READ REQUEST, ABSOLUTE: DCS 23 G Pc; Py; Px ST

Function: This is a request put to the terminal for a RASTER READ REPORT (see below). The start position is specified in the command. The command also specifies the word count. The start position is forced to 16 bit boundary by truncating the start X position to a multiple of 16. The word count specified how many words (2 bytes per word) are to be included in the report.

Pc: Word count.

Py: Start Y (actual start Y used is Py - (Py mod 8).

Px: Start X.

RASTER MEMORY READ REQUEST, RELATIVE: DCS 24 G Pc ST

Function: This is a request put to the terminal for a RASTER READ REPORT (see below). The start position is the current graphics cursor. The command specifies the word count. The start position is forced to 16 bit boundary by truncating the cursor Y position to a multiple of 16. The word count specified how many words (2 bytes per word) are to be included in the report.

Pc: Word count.

RASTER MEMORY READ RESPONSE: DCS 25 G <h..h> ST

Function: This is the response to a RASTER READ REQUEST. It reports the 16-bit values encoded as 4 hex-ASCII values. <h...h> hex-ASCII values.

## AREA FLASH

FLASH AREA, DEFINE: DCS 30 G Pi; Pr; Pyl; Pxl; Pyu; Pxr ST

Function: Define a rectangular region for flashing. The region is specified by identifying the lower left and upper right corner positions.

Pi: Area Id number, 0 - 15.

Pr: Rate: 0=4Hz, 1=2Hz, 2=1Hz, 3=1/2Hz.

Pyl: Y coordinate, lower left corner.

Pxl: X coordinate, lower left corner.

Pyu: Y coordinate, upper right corner.

Pxr: X coordinate, upper right corner.

FLASH AREA , START: DCS 31 G Pi ST

Function: Start a previously defined area flashing.

Pi: Id of flash area, 0 - 15.

FLASH AREA , STOP: DCS 32 G Pi ST

Function: Stop a previously defined area flashing.

Pi: Id of flash area, 0 - 15.

## FAST VECTOR

ENTER FAST VECTOR MODE: DCS 34 G ST <fast vector data>

Function: Enter fast vector mode of operation. See description of this mode in Section 3 - Theory of Operation.

# SECTION 5
# TOUCH CAPABILITIES

## 5.1 INTRODUCTION

The SealTouch infra-red touch sensor, as part of the terminal, is intended to provide a user-friendly input device. The flexibility and apparent simplicity of use make this type of input device the most attractive and least intimidating of all general purpose computer input devices.

**5.1.1 CAPABILITIES:** The SealTouch infra-red touch sensor has three basic methods of reporting user touches. These methods are described below:

The first method simply reports the coordinates of the center of the user's finger. There are four independent modes within this method. These allow reporting of entry, exit, tracking, and multiple touch (error)).

The second method, user defined buttons, detects touches within any rectangular region on the screen and reports back a user-defined message. This method supports several modes of operation. The message can be sent immediately upon touching the region, or only after the finger leaves the screen. This exit mode when used with button highlighting allows accurate selection from tightly spaced regions. A very useful mode automatically draws a boundary around the region on the screen and labels this "button" with the message to be sent or, optionally, with a different label. To facilitate rapid menu changes, the sensor can support multiple pages of button definitions and can switch among them with a single command.

The third method of operation allows the program to overlay the screen with a standard keyboard layout, and to report touches as if they were typed on a keyboard. This allows a user to enter specific information, names, passwords, etc., without having to have a separate keyboard available. This keyboard can be made to appear under software control or, if the software permits, under user control. This keyboard will probably not be acceptable for entering large quantities of data.

It is possible to use all three of these methods concurrently by enabling those desired. There is a hierarchy which causes the on-screen keyboard to override the user-defined-buttons, which in turn override coordinate reporting.

## 5.2 SPECIFICATIONS:

| | |
|---|---|
| Scan rate | 27Hz |
| Touch resolution | 80 x 50 |
| Number of buttons | 121 max |
| Number of menus | 121 max |

**5.3 MAINTENANCE:** No special maintenance procedures are required to keep the touch screen operating properly, except to keep the bezel reasonably clean. Because the bezel and filter are made of plastic, strong solvents should not be used for cleaning.

The tables on the following page give a list of chemicals and the effects they have on the plastic after 7 days immersion at 77° F.

| Class | Name | % | Effect |
|---|---|---|---|
| ACIDS | Acetic Acid | 100 | Dissolves |
| | Acetic Acid | 5 | Negligible |
| | Chromic Acid | 40 | Discolors |
| | Citric Acid | 10 | Negligible |
| | Hydrocloric Acid | 38 | Attacks |
| | Hydrocloric Acid | 10 | Negligible |
| | Nitric Acid | 40 | Attacks |
| | Nitric Acid | 10 | Negligible |
| | Oleic Acid | | Negligible |
| | Sulfuric Acid | 98 | Dissolves |
| | Sulfuric Acid | 30 | Negligible |
| BASES | Ammonium Hydroxide | 28 | Negligible |
| | Sodium Carbonate | 20 | Negligible |
| | Sodium Hydroxide | 60 | Negligible |
| COMMERCIAL PRODUCTS | Cottonseed Oil | | Negligible |
| | Detergent Solution | .025 | Negligible |
| | Kerosene No. 2 D396 | | Negligible |
| | Lacquer Thinner | | Dissolves |
| | Mineral Oil | | Negligible |
| | Olive Oil | | Negligible |
| | Soap Solution | 1 | Negligible |
| | Transformer Oil D1040 | | Negligible |
| | Turpentine D13 | | Crazes |

| Class | Name | % | Effect |
|---|---|---|---|
| INORGANIC COMPOUNDS | Distilled Water | | Negligible |
| | Hydrogen Peroxide | 28 | Negligible |
| | Sodium Cloride | 10 | Negligible |
| | Sodium Hypochlorite | 5 | Negligible |
| ORGANIC COMPOUNDS | Acetone | | Dissolves |
| | Aniline | | Dissolves |
| | Benzene | | Dissolves |
| | Carbon Tetrachloride | | Attacks |
| | Dibutyl Sebacate | | Negligible |
| | Diethyl Ether | | Swells |
| | Dimethyl Formamide | | Dissolves |
| | Ethyl Acetate | | Dissolves |
| | Ethyl Alcohol | 95 | Absorbs 4% |
| | Ethyl Alcohol | 50 | Absorbs 2% |
| | Ethylene Dichloride | | Dissolves |
| | 2-Ethylhexyl Sebacate | | Negligible |
| | Heptane | | Negligible |
| | Isooctane | | Negligible |
| | Methyl Alcohol | | Attacks |
| | Phenol (Aqueous) | 5 | Attacks |
| | Toluene | | Dissolves |

## 5.4 SOFTWARE COMMANDS

**5.4.1 TOUCH MODES:** The terminal supports several basic types of touch reporting. These are discussed in the following paragraphs.

**Entry Mode** sends a report of the coordinates where a finger first breaks the I-R beams. If this is the only active reporting mode, a user could touch the screen and then drag his finger around, but the only report would be the point of first contact.

**Exit Mode** sends a report of the coordinates where a finger last breaks the I-R beams. If this is the only active reporting mode, a user could touch the screen and then drag his finger around, but the only report would be made after the finger is lifted from the screen.

**Track Mode** sends a report of the coordinates of a finger at each change of positions as it moves around the screen. If Entry Mode is not on, it will also report the location of first touch. This mode has no fixed reporting period, but instead reports only when the finger moves.

**Multiple Touch Mode** sends an error report whenever the user touches the screen in more than one place at a time.

**Highlight Touch Mode** reverses the image at the region being touched. This sends no report to the host system, but displays directly. The size of each region is one character. This mode is most useful when used in conjunction with the Exit Mode, as it allows the user to "fine tune" his touch position before activating it.

**Click on Report Mode** causes the terminal to emit an audible click each time a touch report is sent to the host.

The command format for setting these modes is:

CSI > Pm h  to set
CSI > Pm l  to clear

Multiple modes may be set or cleared with one command:

CSI > Pm ; Pm ; Pm ; . . . Pm h  to set
CSI > Pm ; Pm ; Pm ; . . . Pm l  to clear

Where:

**Pm** is an ASCII two-digit number:

**16** Entry Mode
**17** Exit Mode
**18** Track Mode
**19** Multiple Touch Mode
**20** Highlight Touch Mode
**21** Click on Touch Report

All of the above modes, except 20 and 21, are initialized in the cleared state.

All modes will be returned to their default settings by the command:

CSI > 6 D

**5.4.2 TOUCH REPORTS:** Touch reports for the various modes are reported in ANSI compatible formats. All three touch modes have this report format:

CSI > Pl ; Pc Fn

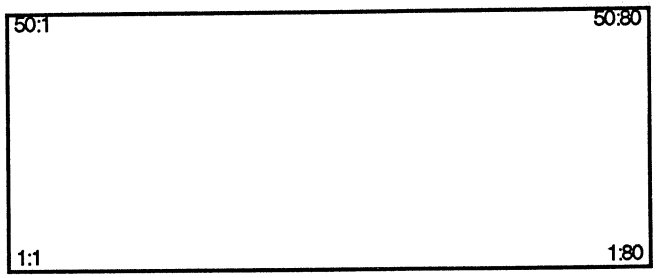
Where:

**Pl** is the vertical touch coordinate in decimal ASCII. range = 1 to 50
**Pc** is the horizontal touch coordinate in decimal ASCII. range = 1 to 80
**Fn** is the final letter of the command which designates the type of report.

**E** Entry Report
**X** Exit Report
**T** Track Report

The Multiple Touch error report has this report format:

CSI > M




**TOUCH REPORTING COORDINATE SYSTEM**

Each touch position corresponds to a character on the display screen.

**5.4.3 TOUCH INQUIRY:** The touch sensor normally sends touch information, depending on which modes are active, on an unsolicited basis; that is, only when the user touches the screen. It is possible for a host system to solicit the touch sensor about current finger position or latest reported finger position.

To request the current finger position send the command:

CSI > 0 n

The system will respond with a standard track mode report if there is something blocking the beams:

CSI > Pl ; Pc T

If there is nothing currently touching the screen it will respond:

CSI > T

To repeat the most recently transmitted touch report from any mode, send the command:

CSI > 1 n

The sensor will respond with the appropriate report type. If no touch report modes are active, it will respond as if the current finger position had been requested.

## 5.5 BUTTON COMMANDS

The most frequent use of a touch sensor is to select something presented on the screen as text or graphics. These touch objects are usually rectangular regions most frequently labeled with text. These touch regions will be referred to as "buttons" in this document. In order to simplify the design of software to utilize buttons, the touch sensor has been provided with capabilities to:

1. resolve touches anywhere within a button and send a user-defined response to the host system;
2. automatically draw boundaries around the region which defines the button and label it with the response or a user-defined label; and,
3. save collections of buttons as "pages" and move quickly among pages with a single command.

Because buttons are frequently relabelled without being repositioned, the touch sensor separates the physical position information from response and label information. Thus, if a row of buttons is reused, it need not be respecified. To manage this separation the sensor uses button numbers in the range from 0 to 120. Each physical button definition must be assigned to a unique button number. All physical buttons are available for use on all pages and do not have a different size or position on different pages. Only those buttons which have a response associated with them on a given page are active (or drawn) when that page is active.

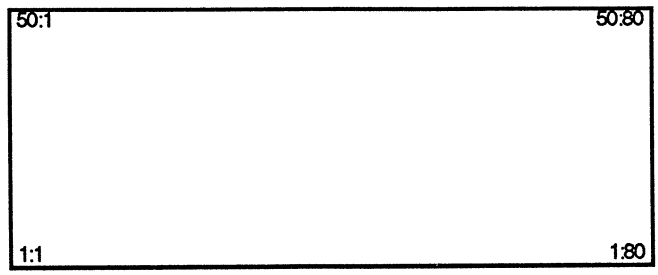
**5.5.1 DEFINE BUTTON LOCATION:** To define the size and position of a rectangular region which is to be used as a button, the host software must send the command:

CSI > Pb ; Pl1 ; Pc1 ; Pl2 ; Pc2 B

Where:

**Pb** is a decimal ASCII integer which specifies the button number.
**Pl1** specifies the row in which the bottom edge of the button appears. Range = 1 to 48
**Pc1** specifies the column in which the left edge of the button appears. Range = 1 to 78
**Pl2** specifies the row in which the top edge of the button appears. Range = 3 to 50
**Pc2** specifies the column in which the right edge of the button appears. Range = 3 to 80
**Pl2** must be at least two greater than **Pl1**. **Pc2** must be at least two greater than **Pc1**. This will allow a single character label within the button boundary. Adjacent buttons may share the same line or column ordinate for the adjacent edge.

This command does not cause the button outline to appear on the display screen. The button will appear on the screen after a response has been defined for it on the current page.




**BUTTON BOUNDARY COORDINATES ON DISPLAY SCREEN**

Button definition commands with out-of-range or missing parameters will be ignored. Subsequent response commands for that button number will also be ignored. The actual touch coordinates which specify edges will be inactive, only the region within the button will generate the response.

When button definitions overlap each other, the button with the higher button number will send its response when the overlapping area is touched. A button may be moved by issuing the command as above with the desired button number and the new position.

**5.5.2 DEFINE BUTTON RESPONSE:** Button responses are completely user-programmable. A button response may be thousands of bytes in length and may contain any binary values desired, including eight-bit values.

In order to assign a response to a button, the button must first be defined as specified above.

The command is:

DCS ~ Pb / Ph Ph Ph . . . Ph ST

Where:

**Pb** is the decimal ASCII button number.
**Ph** is a non-control ASCII code to be returned when the button is touched. Control codes and codes with the eighth bit set may be included by using a backslash preceding a pair of ASCII hexadecimal digits. Control codes may also be included by using ^ preceding the appropriate letter. Other special characters may be included in the response by preceding them with the backslash. Characters which must be preceded with a backslash include: \ : ; ^ | / An example of a button response command to have button 3 respond with "Black:" followed by a carriage return control code would be:

DCS~3/Black\ :^M ST

The ESCape codes in the previous examples are surrounded by spaces for clarity only.

### Button Labels

A button will be labeled with its response, to the degree that the response is displayable text, unless a separate label is specified at the time the button response is defined. The button response command is modified slightly to include a label as follows:

DCS ~ Pb ; Ps / Ph Ph . . . Ph : Pl Pl . . . Pl ST

Where:

**Pl** is any displayable ASCII character except \ : ; ^ | /. Spaces are permitted to separate words in the label. The control character ^M may also be used to force the label to continue on a lower line within the button. Special characters may be included in the label by preceding them with the backslash. Characters which must be preceded with a backslash are: \ : ; ^ |

**Ps** is an optional decimal number (0-63) specifying the label character size and background video state.

Where:

Ps = 0-7, Label character size 0-7 with black background.
Ps = 8-15, Label character size 7 with black background.
Ps = 16-23, Label character size 0-7 with highlighted background.
Ps = 24-31, Label character size 7 with highlighted background.
Ps = 32-39, Label character size 0-7 with vertical hatch background.
Ps = 40-47, Label character size 7 with vertical hatch background.
Ps = 48-55, Label character size 0-7 with horizontal hatch background.
Ps = 56-63, Label character size 7 with horizontal hatch background.

### Local Responses

It may be desirable to have certain buttons send all or part of the response to the terminal itself. With this technique it is easy to set up a button on one page which, when touched, selects another page of buttons, with no interaction with the host.

Using this technique, entire touch driven menu systems can be configured for use with existing programs which have no built-in touch capability. A menu loading program can be run to set up all the required pages of buttons with the responses required by the application program before the application program is started.

To include a local response in a button response command use this format:

DCS ~ Pb / Ph Ph . . Ph | Pr Pr . . . Pr : Pl Pl . . .Pl ST

Where:

**Pr15A** is a non-control ASCII code which will be sent to the display as if it had come from the host when the button is touched. Control codes and codes with the eighth bit set may be included by using a backslash preceding a pair of ASCII hexidecimal digits. Control codes may also be included by using ^ preceding the appropriate letter code. For instance, a carriage return code can be included in the response using ^M. Other special characters may be included in the response by preceding them with the backslash. Characters which must be preceded with a backslash include: \ : ; ^ | /

If no host response is required, the / may be omitted.

### Copying Responses

Frequently, buttons on the different menu pages have identical labels and responses, such as "EXIT" or "HELP." Repeatedly defining these responses for each page on which they are present will use up memory capacity. This can be avoided by reusing a button response that is already stored in the sensor's memory. To copy the response (and label) of a previously defined button, it is necessary to tell the sensor which page number the response is defined on, and which button number it is associated with on that page. The response need not be copied to the same button number. The format of the command is:

CSI > Pnb ; Pob ; Pop C

Where:

**Pnb** is the decimal ASCII new button number. The destination of the copy operation.
**Pob** is the decimal ASCII old button number. The source of the response to be copied.
**Pop** is the decimal ASCII old page number. The page on which the old button response is defined.

This copy requires no additional memory for the first 30 copies. If more than 30 copies are required, copies 32-62 should be made referencing copy 31, or better yet, redefine the response and make further copies of it. If the response for the old button is later changed, it will have no effect on the new button.

**5.5.3 MULTI-STATE BUTTONS:** Button responses can be programmed in such as way that each successive touch will send a different host and local response from a list of sequential states. In addition, each state may relabel the button if desired, may highlight the button, or change the button video state. Up to 16 states may be defined for any button.

When a button response is copied to another page of buttons, the state of the button and the highlighting will be carried with the response, regardless of which button outline the response is associated with on that page. If a response is used more than once on a single page, it will not highlight or relabel the untouched copies.

The format of the command to set up a multi-state response is:

DCS ~Pb ; Ps0 ; Ps1 ; ... ; Psn
/ Ph0 | Pr0 : Pl0
/ Ph1 | Pr1 : Pl1
.
.
.
/ Phn | Prn : Pln ST

Where:

**Pb** is the button number on the definition page
**Psx** is the label type for state x
**Phx** is the host response for state x
**Prx** is the local response for state x
**Plx** is the label for state x, may be : to use label of x-1
**n** is 0 to 15

**5.5.3.1 BEHAVIOR OF MULTI-STATE BUTTONS:** The button will initially be labelled with the label specified by state 0. When it is actuated, the host and local responses used will be those of state 0. The button will then be relabeled with the next states label. When the final state is actuated, the state will return to 0.

If no label is supplied on any given state, the host response will be used to label the button in that state. If no change is desired in the label, use a double colon to specify the label on each state that should not change.

If a label type is specified for a corresponding state, it can be used to change the highlighting, and of a button as well as the size of the label in each state that the button moves through.

**5.5.4 BUTTON MANIPULATION COMMANDS:**

**Actuate Button Command** allows the host computer system to "press" one of the buttons on the current page. The format for the command is:

CSI > Pb A

Where:

**Pb** is the decimal ASCII button number on the current page.

**Set Button State** command allows the host computer system to set the state of a multi-state button on the definition page (not current page). The format of the command is:

CSI > Pb ; Ps A

Where:

**Pb** is the decimal ASCII button number
**Ps** is the decimal ASCII state, range 0-15

**5.5.5 BUTTON MODES:** The button functions have several options which are controlled by modes. The individual modes are described in the following paragraphs.

**Buttons Enabled** mode allows all the buttons on the current page to be enabled or disabled at once. This mode defaults to the set state.

**Button Auto-Draw** mode automatically draws the outline and labels each button as they are defined on the current page. When page changes are made, it will remove all old buttons from the screen and draw all the new ones automatically. This mode defaults to the set state.

**Buttons Entry/Exit** mode determines whether buttons will send their response when first touched, or when the finger is removed from the screen. When in Entry mode, a button will only be actuated if it is touched when the finger was not already on the screen. If the finger is dragged across the screen, buttons will not be actuated. In Exit mode, the finger must leave the screen directly from the button in order to actuate. This Button Exit mode is primarily intended for tightly spaced buttons or list style menus where the user would need some visual feedback to ensure correct selection. If this mode is cleared (to Exit mode) the highlight mode should be set. This mode defaults to the set (entry) state.

**Highlight Button Touches** mode will cause the outline of a button when touched to become highlight. The button will return to normal when the finger leaves the button by lifting or sliding. When in Button Entry Mode, highlighting indicates button actuation. When Buttons are in Exit mode, a button will highlight when touched to indicate that it will actuate if the finger is lifted. This mode is set by default.

**Button Beep** mode causes an audible signal to be generated when a button's response is sent. This mode is set by default.

**Clear Screen on Page Change** mode erases the entire screen when a page change is made. Note that issuing a command to set the page to the current page has no effect. This mode avoids erasing individual buttons. This mode is set by default.

**Button Zero Highlight & Border Suppress** mode is intended for use with button zero on a given page being a "background button." By a "background button" we mean a button defined to be the area of the entire screen, or at least surrounding all of the higher numbered buttons. In this mode, button zero does not get a border drawn nor get highlighted when touched. It defaults to the cleared state.

Button Auto-Draw, Clear Screen on Page Change and Button Zero Highlight & Border Suppress require a page change before they will take effect.

**Button Label Pack Mode.** Multi-line button labels are normally given normal text line spacing. It may be desirable to place button label lines as close together as possible. When enabled, this mode will pack button label lines very close together (actual distance depends on label size used). This mode is disabled by default.

The command format for setting these modes is:

| | |
|---|---|
| CSI > Pm h | to set |
| CSI > Pm l | to clear |

Multiple modes may be set or cleared with one command:

CSI > Pm ; Pm ; Pm ; . . . Pm h  to set
CSI > Pm ; Pm ; Pm ; . . . Pm l  to clear

Where:

**Pm** is an ASCII two-digit number:

**23** Buttons enabled
**24** Button auto-draw
**25** Buttons entry/exit
**26** Highlight button touches
**27** Button Beep
**28** Clear screen on page change
**29** Button zero highlight & border suppress
**40** Button label pack/normal spacing

All touch modes may be reset to their default condition with the command:

CSI > 6 D

**5.5.6 DRAWING AND ERASING BUTTONS:** Buttons are usually drawn on the screen automatically by the touch sensor. This operation may be overridden by clearing the auto-draw mode. If you subsequently want the buttons to appear, merely setting auto-draw will not draw the buttons until the next page change.

The command required to draw the buttons on the current page is:

CSI > 3 D

Similarly, to erase the buttons from the screen only (they will still be operational) send the command:

CSI > 4 D

**Erasing Buttons**

All buttons, responses, labels and pages may be erased with:

CSI > B

This effectively reinitializes the button section of the touch sensor.

**5.5.7 BUTTON MENU PAGES:** The touch sensor is initialized with only a single page allocated for button responses. This is page number zero. If the user never selects another page, buttons will be drawn at the time they are defined and removed when they are erased. This operating method may suffice for some applications. If the user doesn't mind re-transmitting button responses each time different buttons are required, then he need never use another page. For more complex applications, or those where the fastest response times are desired, the use of multiple pages of button responses will help.

**Selecting A Page**

At any point in time the sensor has a "current" page and a "definition" page. The current page reflects that page which is drawn on the screen, and whose responses are active for touches. The definition page, which may be the same as current, is the page on which defining actions such as erase or copy response take place. This allows touch actions on the current page to help define or change buttons elsewhere.

To set the current page and definition page to the same number, use the command:

CSI > Pc P

Where:

**Pc** is decimal ASCII page number from 0 to 120.

To set the definition page to a different page than the current page, send the command:

CSI > Pc ; Pd P

Where:

**Pd** is decimal ASCII page number from 0 to 120.

Please note that dynamic space allocation in the touch sensor allocates space for all pages between 0 and the highest page number used. To conserve space, use sequential page numbers starting at zero. The amount of memory allocated to each page is proportional to the maximum button number that has been defined.

**Clearing Pages**
The current definition page can have all of its button responses erased by a single command:

CSI > 2 D

Note that if the page you are erasing happens to be the highest page number in use, the memory used by that page will be recovered for re-allocation.

You may erase all pages (but not the button definitions) with the command:

CSI > 1 D

**5.5.8 BUTTON RESOURCE INQUIRY:** The button facilities use internal memory to save each location and response. Dynamic allocation is used to allow trading off numbers of buttons, pages, and length of button responses. In order to better manage the memory resource in a complex operating environment, you can command the sensor to send utilization information back. The lowest unused button number, the highest used button number, the next higher unused page number, and the remaining memory size (free space) may all be requested. This allows each new button to be assigned to the next available button number, and each new page to be assigned to the next page. In this manner, applications which may not communicate can have their button definitions co-reside in the touch sensor.

The commands are formatted like this:

CSI > Ps n

Where:

**Ps** is: **7** for next page
**6** lowest unused and highest used buttons
**5** for free space

The response will return as:

| | |
|---|---|
| CSI > Pn p | for next page |
| CSI > Pl ; Ph b | for next button |
| CSI > Ps f | for free space |

Where:

**Pn** is a Decimal ASCII number: 0 to 120 for pages
**Pl** is a Decimal ASCII number: 0 to 120 for lowest unused button number
**Ph** is a Decimal ASCII number: 0 to 121 for highest used button number +1
**Ps** is a Decimal ASCII number: 0 to 53115 bytes for free space

The amount of space you require may be calculated as follows:

bytes required = B(4 + 2P) + R – 96

Where:

**B** is number of buttons allocated = 16 (highest_used_button/16 + 1)
**P** is number of pages (next page)
**R** is number of bytes used for storing responses and labels

R 2 NR + TRL + NLR + TLRL+NL+TLL+N8C + 3 DCL + MSHR

Where:

**NR** is number of responses (even if only local)
**TRL** is total response length in bytes
**NLR** is number of local responses
**TLRL** is total local response length in bytes
**NL** is number of labels
**TLL** is total label length in bytes
**N8C** is number of response codes with the 8th bit set
**DCL** is number of double colon labels
**MSHR** is number of multi-state host responses

Copied responses require no memory for the first 30 copies. After that, additional allocations will occur for each copy.

**5.5.9 POP-UP MENUS:** The use of pop-up (pull-down) menus was popularized by Apple Computers with the introduction of the Macintosh. This style of menu utilizes a row of menu titles located along the top or bottom of the screen. When one of these title buttons is touched a secondary menu appears on the screen adjacent to the title. This secondary menu is a list of items each of which will highlight as the finger is dragged across it. When the desired item is highlighted, lifting the finger off the screen will send the associated response to the host and immediately remove the secondary menu from the screen.

This special type of touch operation uses a command to facilitate these pop-up menus. Even with the command, quite a bit of set-up is required. A separate page must be defined for each secondary menu, and the individual items must have their responses defined. Each item must be assigned as if it were a separate button, starting with button number one and continuing sequentially to no higher than 23. Corresponding button outlines need not be defined. No more than 15 pop-up menu lines can be displayed

The procedure for defining a page to be used as a pop up menu is to set the desired page to the definition page, which must be different than the current page, which must be a normal button page.

CSI > Pcurrent ; Pdefinition M

Now define all the items on the pop-up using "button numbers" 1 to n and the command for assigning button responses or copying responses.

The box which surrounds the items in the list is derived from the number of items and the length of their labels. The box and all the items in it are positioned by the invoking command, or automatically from the position of the last button activated.

The position of the menu is determined by the base coordinate. If the base coordinate is in the upper half of the screen, the box will be drawn below. If the base coordinate is in the bottom half of the screen, the box will be drawn above it.

If the items in the resulting pop-up menus are too close to each other for good legibility, you may wish to use even numbers only, starting with "button number" 2.

The primary menu titles should be defined as buttons and have local responses of the format:

CSI > Pp M or: CSI > Pp ; Py ; Px M

Where:

**Pp** is the decimal ASCII page number of the desired secondary menu page.
**Py** is the Top or Bottom coordinate of the secondary menu
**Px** is the left coordinate of the secondary menu

If the positioning coordinates are omitted from the command, the base coordinates will be derived from the last button which was activated. If that button is in the top half of the screen, the coordinates will be taken from the bottom edge of the button. Conversely, if it is in the bottom half of the screen, the coordinates will be taken from the top edge.

Although the normal mode of operation leaves the primary menu visible (but inactive), there is a mode command to cause the primary menu to disappear while the secondary menu is active:

CSI > 30 h  to set
CSI > 30 1  to clear

If a response is assigned to "button number" 0 on a pop-up page, that response will activate if the user's finger is outside the secondary menu when it leave the screen.

**5.5.10 BUTTON TIME-OUT:** In some applications it is desirable to trigger action based upon lack of user activity. Such a situation might arise where the user walks away and the system should return to some main menu.

To simplify the solution to this problem a "time-out" function has been added to the touch sensor. One button on the current page may be assigned a time, in seconds, which will be used to measure inactivity (no finger motion). If the assigned number of seconds pass with no activity, the touch sensor behaves exactly as though the user had activated the button to which the "time-out" was assigned.

The command for assigning a "time-out" to a button is:

CSI > Pb; Pt B

Where:

**Pb** is the button number
**Pt** is the time in seconds (Range = 1 to 255) (0 means disable)

A page change will disable the time-out.

## 5.6 ON-SCREEN KEYBOARD

Many applications where touch sensors are used have only occasional need to input alpha-numeric information. The touch sensor incorporates a full ASCII keyboard capability for meeting these limited requirements. This on-screen keyboard doesn't waste any of the user-definable buttons, or message storage space and requires no extensive software set-up. The keyboard may be called up by either a command from the host or a touch from the user.

The keys on the keyboard operate in entry mode. The shift key and the control key operate a little differently than a standard keyboard because of the single touch restrictions, so they must be touched first, and then the desired key may be touched. The shift or control key will light until the next key is touched. The caps lock key will toggle when touched. Function keys are included on a second keyboard callable from the main keyboard.

**5.6.1 KEYBOARD MODES:** The Keyboard Enabled mode is defaulted to ON at power up and allows the user to call up the keyboard at will. To turn the keyboard on or off, the user touches the extreme lower right-hand corner of the screen, where a very small touch area (2 x 2) is set-up to detect entry, only. This area remains useful for normal tracking, exit and button exit responses. When this mode is disabled there is nothing special about the area.

The Keyboard On mode allows host software to anticipate the user's need for the keyboard and bring it up on the screen, or, by clearing the mode, remove it from the screen. This mode can be set or cleared by the user if the Keyboard Enable mode is set.

The command format for changing these modes is:

CSI > Pm h to set
CSI > Pm l to clear

Where:

**Pm** is an ASCII two-digit number:
**33** Keyboard Enables
**32** Keyboard On

## 5.7 TOUCH PROTECT

The touch screen sensitivity may be adjusted for harsh environments to exclude any touch which might be considered too small or to large. Additionally, a delay can be added to require more than just a quick hit.

This feature reduces undesired actuations or so-called "false hits." A false hit might be caused by objects laying on , dropped onto or bounced off of the touch panel, or by the touch panel being bumped or brushed by a person, an animal or an insect.

To avoid accidental touch reports, the minimum and maximum size of objects that will trigger the touch sensor may be programmed. A delay between the time a touch is sensed and the time the report is triggered may also be programmed.

To program the touch sensitivity, send the command:

CSI>Pmax;Pmin;Pd S

Where:

**Pmax** is the maximum-sized object that will trigger the touch sensor. Range 2-80. Each increment equals .192 inches. The maximum size must always be at least two increments larger than the minimum.
**Pmin** is the minimum-sized object that will trigger the touch sensor. Range 0-325 Each increment equals 0192 inches.
**Pd** is the delay in increments of 0.04 seconds. Range 0-50.

*Examples:*

CSI>10;3;35 S Maximum size for valid touch is 1.92 inches, minimum size for valid touch is .576 inches; delay is 1.4 seconds.
CSI3;0;0 S Maximum size is .576 inches, minimum size is undefined; there is no delay.

*NOTE:* Spaces shown in command are for clarity only. No spaces are inserted when command is sent.

## 5.8 SELF-CHECK FUNCTIONS

The touch sensor contains a microcomputer which can perform several self-checking functions. To have the unit perform a self-check and return its status, send the command:

CSI > 4 n

The sensor will check the operation of all infra-red beams, test the system ram, and check the program memory check-sum. This test uses a special technique to test all ram while not disturbing the contents. This takes a little longer than a simpler destructive memory test, but leaves all state information, communication queues, and button definitions intact. This will take either four seconds, or sixteen seconds, depending on the size of the ram chip installed. During this period, the device will

not respond to touches or computer commands (and none should be attempted or they will be lost). The screen should not be touched during the test or erroneous results for the I-R beams will result. The sensor will return the results in this format:

CSI > Pf c

Where:

**Pf** is a decimal ASCII digit which should be 0 if all systems are working properly. The low order bits each represent one of the systems.

**1** Checksum failure
**2** Ram test failure
**4** I-R beam failure

Each error found by the self-test will increment an error counter which may be interrogated and cleared with the command:

CSI > 2 n

Which will respond with:

CSI > Pn e

Where **Pn** is a decimal number: Range = 0 to 65535

If the sensor indicates that an infra-red beam has failed, you may ask it to report which component has failed by sending it the command:

CSI > 3 n

The sensor will respond with:

CSI > Pf ; Pf ; Pf ; . . . Pf a

Where:

**Pf** is a decimal ASCII integer that identifies which component has failed. Any number of failure codes may be sent. Component identification codes begin with 00 for the left most LED on the X axis. Even numbers identify the LED's and odd numbers identify the phototransistors. Numbers count upward to the right in the X axis and upward in the Y axis. There is no gap in numbering between axes. When mode 34 is set, the sensor may respond with this report if a beam failure is detected during operation.

*NOTE*: The touch sensor performs a self-test at power-up and audibly returns the result using the beeper. The code that is sent is the same as for CSI>4n. One beep represents a checksum failure, three beeps represent a checksum failure and a RAM text failure, etc.

**5.8.1 CONFIGURATION INQUIRY:** The touch sensor supports two inquiries for determining its configuration. The dimensions of the touch array may be requested by sending:

CSI > 8 n

This will return: CSI > Py ; Px d

Where:

**Py** is a decimal ASCII integer which is the maximum Y axis coordinate which can be returned by the sensor. (Ymax)
**Px** is a decimal ASCII integer which is the maximum X axis coordinate which can be returned by the sensor. (Xmax)

Coordinates returned by the sensor range from one to the maximums returned by this command.

The current version of the software may be requested by sending:

CSI > 9 n

Which will return: CSI > Pv v

**Pv** is a decimal ASCII integer which reflects the version. The number 10 implies 1.0.

# SECTION 6
# RS-485 MULTI-DROP PROTOCOL

## 6.1 DESCRIPTION

The scheme is a variation of a token-pass network. Each terminal in the system is assigned an I.D. The I.D. is a number, 0 to 255. I.D. = 0 is used as a "Null I.D." in networked systems. A terminal may be designated as a Receiver and/or Transmitter by the host using the ANSI 3.64 commands, below.

A Receiver responds to all host transmissions; it displays displayable ASCII characters and manipulates the data on its screen according to ANSI commands sent by the host. Non-Receivers ignore such normal data transmissions from the host. Non-Receivers listen only for the network commands listed below.

A Transmitter may be one of two kinds. It may be designated as a continuous transmitter and may transmit regardless until the host designates another Transmitter. The Transmitter may also be designated as a short-span transmitter. A short-span transmitter transmits all current data destined for the host, and then selects another Transmitter by terminating its own transmission with the Transmit Control Command (CSI Pid |), where Pid is the value from the next I.D. field in the terminal's communication set-up screen. Note that if the next I.D. field is 0 (zero), no terminal is selected as the Transmitter.

A continuous Transmitter may be forced into a temporary short-span mode if desired by sending it the short-span designator command with an I.D. of zero. The transmitter will then finish its transmission and terminate it with a short-span designator command (CSI 0 |). This allows the terminal to decide the proper place to terminate transmission, and does not automatically designate another terminal as Transmitter.

Networking is enabled by the "NETWORK" field in the communications set-up screen. Networking is independent of the I/O type (RS232, RS485, etc.) but only makes sense for RS485.

The "I/O TYPE" field in the communications set-up screen allows selection of RS485. 1 or 2 channel selection corresponds to whether the terminal's receive and transmit lines (assumed to be connected to the network bus) are one and the same.

### 6.1.1 SET-UP FIELDS

| | |
|---|---|
| I.D.: | 0-255 |
| Next I.D.: | 0-255 |
| I/O Type: | RS232/RS485 (1 channel or 2 channel) |
| Network: | Yes/No |

### 6.1.2 HOST COMMANDS

RECEIVE CONTROL: CSI Pid {

Function: Pid = 0: All terminals receive. Pid = 1-255: Only terminal Pid receives.

TRANSMIT CONTROL, CONTINUOUS: CSI Pid }

Function: Pid: Terminal Pid is the designated continuous transmitter.

TRANSMIT CONTROL, SHORT-SPAN: CSI Pid |

Function: Pid: Terminal Pid is the designated short-span transmitter.

In addition, there is a command which will enable or disable multidrop altogether and define the multidrop ID and next ID.

CSI < Penb; Pid; Pnxtid @

where

Penb = 0 disables multidrop
1 enables multidrop

Pid is the Multidrop ID for the device

Pnxtid is the ID of the device to be used in the short-span termination sequence.

# APPENDIX A
# TERMINAL SPECIFICATIONS

| | |
|---|---|
| Overall Dimensions: | 11.64" W x 10.58" H x 3.15" D |
| Viewing and Touch Area: | 7.68" W x 4.84" H |
| Display: | |
| Technology | Electroluminescent |
| Pixel Matrix | 640 x 400 |
| Pixel Aspect Ratio | 1:1 |
| Weight: | 13.6 lbs. |
| Operating Temperature: | 0 to 45°C |
| Storage Temperature: | -45 to 85°C |
| Humidity: | 0 to 100% |
| Power Requirements: | 60Hz, 120VAC @ .5A Maximum, .3A Typical |
| Ratings: | ST3220: UL listed, NEMA 4 & 12<br>ST3225: NEMA 4 & 12<br>ST3420: NEMA 4 & 12 Front Panel |
| Touch Modes: | X-Y Coordinate Reporting<br>Enter, Exit, Track<br>Up to 120 pages of up to 120 buttons<br>Pop-Up Menus<br>On Screen Keyboards (ABC, QWERTY, Function Pad)<br>Touch Sensitivity |
| Memory: | Two pages of video memory. |
| Command list storage: | 16K bytes (Standard)<br>64K bytes (Optional)<br>Non-Volatile |
| Interfaces: | RS-232C, RS-422, RS-485 (MultiDrop)<br>Auxiliary Serial Printer<br>Auxiliary IBM AT Mechanical Keyboard |
| Limited Warranty: | 1 Year |

# APPENDIX B
# CONNECTOR PIN ASSIGNMENTS

## HOST INTERFACE CONNECTION:

*For the tables below, "high" refers to a voltage of about +12V and "low" a voltage of about –12V on the RS232 cable.*

| PIN | SIGNAL | MNEMONIC | DESCRIPTION |
|---|---|---|---|
| 1 | Chassis Ground | CGND | Ground signal to which cable shield should be connected. |
| 2 | Transmit Data | TxD | (Output) Transmits serial data. Held low during idle. If modem control mode is enabled, data is only sent when DSR, CTS and DCD are high. |
| 3 | Receive Data | RxD | (Input) Receives serial data. |
| 4 | Request to Send | RTS | (Output) Used as M4ST busy status. Held low when M4ST is busy. |
| 5 | Clear to Send | CTS | (Input) Used as a host busy status. M4ST will not transmit if CTS is low. |
| 6 | Data Set Ready | DSR | (Input) Not currently used. |
| 7 | Signal Ground | SGND | Ground reference for RS-232C and modem control signals. |
| 8 | Data Carrier Detect | DCD | (Input) Not currently used. |
| 12 | Alternate Receive | ALT RCV | (Input) Alternate receive data for optional communications protcol (RS485 ) |
| 13 | /Alternate Receive | /ALT RCV | (Input) Alternate receive data for optional communications protcol (RS485). Inverted. |
| 20 | Data Terminal Ready | DTR | (Output) Follows RTS. |
| 24 | Alternate Transmit | ALT XMT | (Output) Alternate transmit data for optional communications protcol (RS485) |
| 25 | /Alternate Transmit | /ALT XMT | (Output) Alternate transmit data for optional communications protcol (RS485). Inverted. |

## PRINTER INTERFACE CONNECTION:

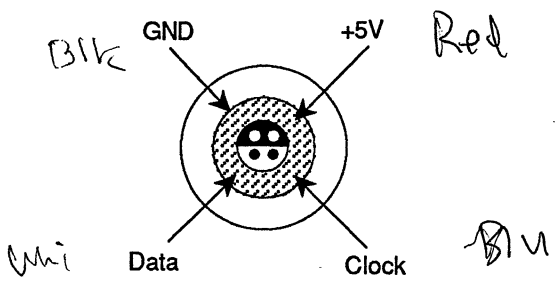
*For the tables below, "high" refers to a voltage of about +12V and "low" a voltage of about –12V on the RS232 cable.*

| PIN | SIGNAL | MNEMONIC | DESCRIPTION |
|---|---|---|---|
| 1 | Chassis Ground | CGND | Ground line to which cable shield should be connected. |
| 2 | Transmit Data | TxD | (Output) Transmits serial data. Held low during idle. If hardware handshake is enabled, data is only sent when DSR is high. |
| 3 | Receive Data | RxD | (Input) Receives serial data. |
| 4 | Request to Send | RTS | (Output) High when terminal ready to transmit. |
| 5 | Data Terminal Ready | DTR | (Output) Follows RTS. No DTR handshaking supported. |
| 6 | Data Set Ready | DSR | (Input) If modem control mode is enabled, data is sent by the terminal when this line is high. Ignored otherwise. |
| 7 | Signal Ground | SGND | Signal Ground reference. |

## KEYBOARD INTERFACE CONNECTION:

| PIN # | NAME | DESCRIPTION |
|---|---|---|
| 1 | +5V | Supplies power to the keyboard. |
| 2 | GND | Signal ground for the keyboard. |
| 3 | DATA | Serial signal which defines the scan code. |
| 4 | CLOCK | Used to latch the incoming data. |

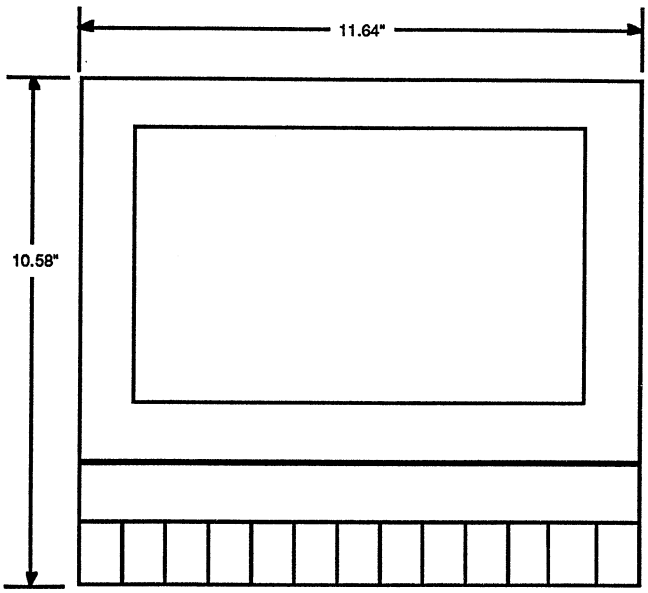
**Outside View**



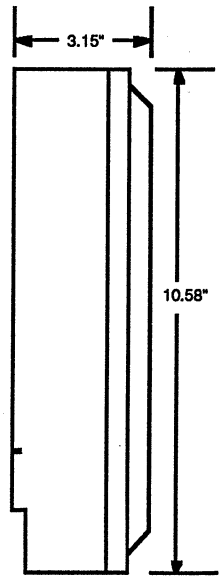

**ST3220 Keyboard Pinout**

# APPENDIX C
# MECHANICAL DRAWINGS
## ST3220 / ST3225

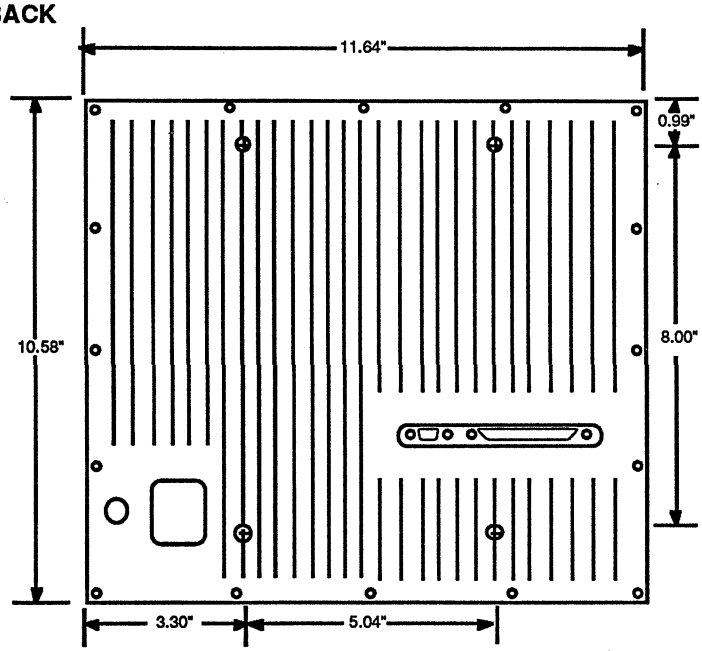
**FRONT**




**SIDE**

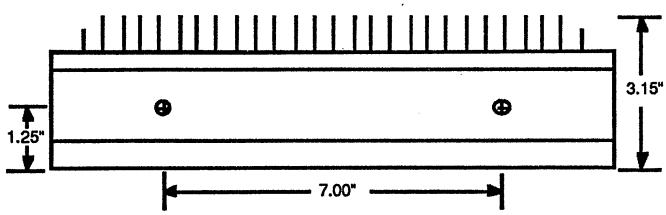



**BACK**



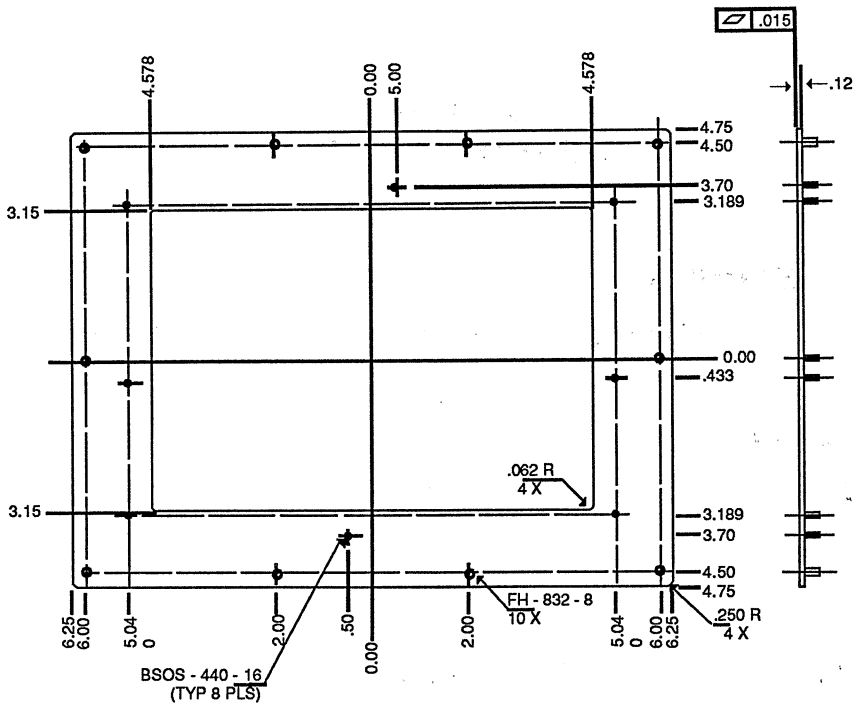

Mounting holes are
1/4 – 20 UNC .5"

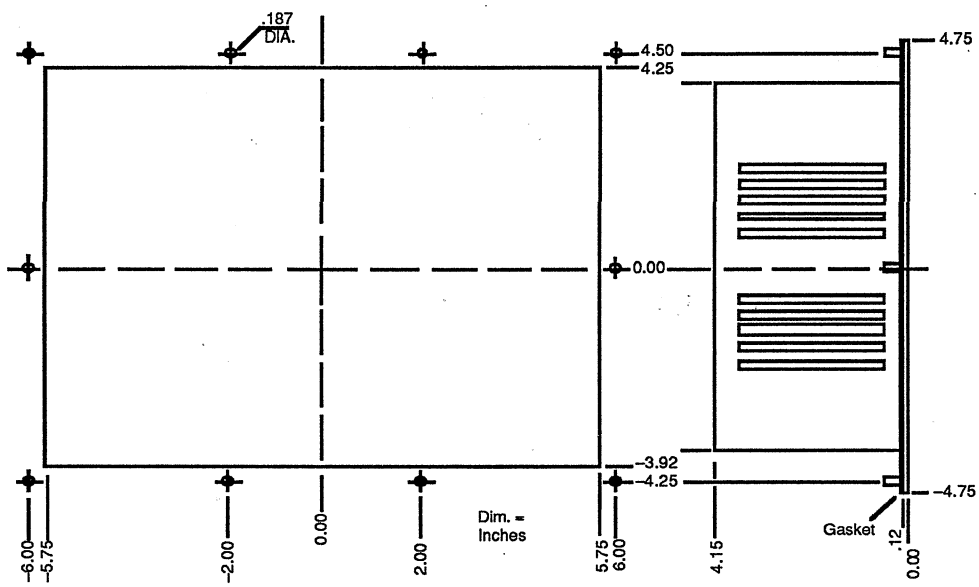
**BOTTOM**




*NOTE:*
When mounting the ST3220, it is recommended that both the rear and bottom mounting holes be used. When using this method, care must be taken not to block vertical airflow through the cooling fins.

## ST3420 – Front Panel Mount




**Front Panel Cutout** **Side**

# APPENDIX D
# DEFAULT SETTINGS

## DEFAULT SETTINGS

| | | |
|---|---|---|
| ALPHA CHARACTER ATTRIBUTES | Inverse: | Off |
| | Italic: | Off |
| | Underline: | Off |
| ALPHA CHARACTER SIZE | Size: | 1 |
| ALPHA CURSOR | Style: | Box |
| | Visible: | Yes |
| | Location: | Home (row 1, column 1) |
| ALPHA LINE | Auto-Wrap: | Off |
| | Newline: | On |
| ALPHA, OTHER | Write Mode: | Jam |
| | Character Set: | Standard |
| | Character Size: | 1 |
| GRAPHICS CHARACTER ATTRIBUTES | Character Size: | 1 |
| | Reverse Video: | Off |
| | Italics: | Off |
| | Underline: | Off |
| | Character Set: | Standard |
| GRAPHICS CONTROL | Window Size: | Full Screen |
| | Window Origin: | Bottom, Left of Alpha Page |
| | Window Type: | Non-Alpha |
| | Clipping: | On |
| GRAPHICS CURSOR | Style: | Cross |
| | Visible: | No |
| | Location: | (0,0) |
| GRAPHICS OBJECT ATTRIBUTES | Line Style: | Solid white |
| | Fill Style: | Solid white |
| | Write Mode: | Jam |
| GRAPHICS STRING | String Wrap: | Off |
| | Control Chars.: | Display |
| | Rotation: | 0° |
| | Path: | Left to right |
| HOST COMMUNICATION | IO Type: | RS232 |
| | Baud: | 9600 |
| | Data bits: | 8 |
| | Stop bits: | 1 |
| | XON/XOFF: | Yes |

| | | |
|---|---|---|
| KEYBOARD | Auto-Repeat: | On |
| | Key Mapping: | On |
| NETWORK: | Enable: | No |
| | ID: | 0 |
| | Next ID: | 0 |
| PRINTING | Auto-Print: | Off |
| | Print Control: | Off |
| | Print Extent: | Screen |
| | Print Input: | Disabled |
| | Print Terminator: | Form Feed |
| SCREEN SAVER | Enable: | No |
| | Time-Out: | 10 minutes |
| SCROLL MARGINS | Top: | 1 |
| | Bottom: | 24 |
| TABS | Columns: | 9, then every 8th column |
| TERMINAL MODES | Term: | VT100 |
| | 7/8 Bit: | 7 bit |
| TOUCH | X-Y Touch: | No mode enabled |
| | X-Y Highlight: | Yes |
| | X-Y Beep: | No |
| | Buttons Enabled: | Yes |
| | Button Highlight: | Yes |
| | Button Beep: | Yes |
| VIEWPORTS | Viewport 0: | Alpha absolute,offset=(0,0), size=full screen |
| | Viewport 1: | Graph absolute,offset=(0,0), size=0 |

# APPENDIX E
# COMMAND COMPARISON

## COMMAND COMPARISON CHART
(X = Featured)

| COMMAND | | VT-220 | VT-100 | DeeCO ST3220 |
|---|---|---|---|---|
| ALPHA CONTROL | Alpha Character Size | | | X |
| | Character Set Select | | | X |
| | Alpha Write Mode | | | X |
| AREA FLASH | Flash Area, Define | | | X |
| | Flash Area, Start | | | X |
| | Flash Area, Stop | | | X |
| AUXILLARY MEMORY | Auxilliary Memory Type | | | X |
| | Compute/Store Checksum | | | X |
| CHARACTER ATTRIBUTES | All Attributes Off | X | X | X |
| | Bold On | X | X | |
| | Underline On | X | X | |
| | Reverse Video On | X | X | X |
| | Bold Off | X | | |
| | Underline Off | X | | |
| | Blink Off | X | | |
| | Reverse Video Off | X | | X |
| | Multiple Attribute On/Off | | | X |
| CHARACTER DEFINITION | Define Alternate Character | | | X |
| CHARACTER PROTECTION | No Protect | | X | |
| | Bold Protect | | X | |
| | Underline Protect | | X | |
| | Blink Protect | | X | |
| | Reverse Protect | | X | |
| | All Protect Off | | X | |
| | Erase Attributes Off | X | X | |
| | Protect Character | X | | |
| | Un-Protect Character | X | | |
| CHARACTER SET SELECTION | Lock Shift G0 | X | X | |
| | Lock Shift G1 | X | X | |
| | Single Shift 2 | X | X | |
| | Single Shift 3 | X | X | |
| | Lock Shift, G1, rt | X | | |
| | Lock Shift, G2 | X | | |
| | Lock Shift, G2, rt | X | | |
| | Lock Shift, G3 | X | | |
| | Lock Shift, G3, rt | X | | |
| | G0=UK | X | X | |
| | G0=US | X | X | |
| | G0=DEC Special | X | X | |
| | G0=xxx | X | X | |
| | G0=DEC Supplemental | X | | |
| | G1=UK | X | X | |
| | G1=US | X | X | |
| | G1=DEC Special | X | X | |
| | G1=xxx | X | X | |
| | G1=DEC Supplemental | X | | |
| | G2=US | X | | |
| | G2=DEC Special | X | | |
| | G2=DEC Supplemental | X | | |

## COMMAND COMPARISON CHART (con't.)

(X = Featured)

| COMMAND | | VT-220 | VT-100 | DeeCO ST3220 |
|---|---|---|---|---|
| CHARACTER SET SELECTION (con't.) | G3=US | X | | |
| | G3=DEC Special | X | | |
| | G3=DEC supplemental | X | | |
| CURSOR POSITIONING | Cursor Up | X | X | X |
| | Cursor Down | X | X | X |
| | Cursor Forward | X | X | X |
| | Cursor Backward | X | X | X |
| | Cursor Position | X | X | X |
| | Cursor Home | X | X | X |
| | Horizontal and Vertical Position | X | X | X |
| | Index | X | X | X |
| | Reverse Index | X | X | X |
| | Next Line | X | X | X |
| | Save Cursor | X | X | |
| | Restore Cursor | X | X | |
| DATA/DISPLAY LISTS | Open Display/Data List | | | X |
| | Close List | | | X |
| | Execute Display List | | | X |
| | Append Data List | | | X |
| | Read Data List | | | X |
| | List Report | | | X |
| | Clear All Lists | | | X |
| DOWN-LOAD CHARACTER | Define Character | X | | |
| EDITING | Erase Line From Cursor | X | X | X |
| | Erase Line To Cursor | X | X | X |
| | Erase Line | X | X | X |
| | Erase Screen From Cursor | X | X | X |
| | Erase Screen To Cursor | X | X | X |
| | Erase Screen | X | X | X |
| | Delete Character | X | | |
| | Insert Line | X | | |
| | Delete Line | X | | |
| | Insert Spaces | X | | |
| EIGHT BIT CONTROL | Transmit 7-Bit Equivalent | X | | X |
| | Transmit 8-Bit Control | X | | X |
| FAST VECTOR | Enter Fast Vector Mode | | | X |
| GRAPH COMMAND CONTROL | Graphic Window Limits | | | X |
| | Graphic Window Origin | | | X |
| | Graphic Window Type | | | X |
| GRAPHIC ATTRIBUTES | Character Attribute Select | | | X |
| | String Attribute Select | | | X |
| | Line Style Attribute Select | | | X |
| | Fill Style Attribute Select | | | X |
| | Graph Write Mode Select | | | X |
| GRAPHIC OBJECT | Clear Graphic Window | | | X |
| | Set Current Position | | | X |
| | Vector Draw, Absolute | | | X |
| | Vector Draw, Relative | | | X |
| | Circle Draw, Absolute | | | X |
| GRAPHIC OBJECT (con't.) | Circle Draw, Relative | | | X |
| | Arc Draw, Absolute | | | X |
| | Arc Draw, Relative | | | X |
| | Block Fill Draw, Absolute | | | X |
| | Block Fill Draw, Relative | | | X |
| | Concave Polygon Fill, Absolute | | | X |
| | Concave Polygon Fill, Relative | | | X |
| | Convex Polygon Fill, Absolute | | | X |
| | Convex Polygon Fill, Relative | | | X |
| | Graphic String Draw, Absolute | | | X |
| | Graphic String Draw, Relative | | | X |

## COMMAND COMPARISON CHART (con't.)

(X = Featured)

| COMMAND | | VT-220 | VT-100 | DeeCO ST3220 |
|---|---|---|---|---|
| KEYPAD SELECTION | Numeric Keypad | X | X | X |
| | Application Keypad | X | X | X |
| LINE ATTRIBUTES | Double Height Top | X | X | |
| | Double Height Bottom | X | X | |
| | Single Width | X | X | |
| | Double Width | X | X | |
| OTHER | Screen Save Timeout | | | X |
| | Enter C2 Mode | | | X |
| | Return to ST3220 Mode | | | X |
| | Erase Graphic Window | | | X |
| | Select User Input | | | X |
| PRINTING | Auto Print On | X | X | X |
| | Auto Print Off | X | X | X |
| | Print Control On | X | X | X |
| | Print Control Off | X | X | X |
| | Print Cursor Line | X | X | X |
| | Print Screen | X | X | X |
| | Print Buffer Response | | | X |
| | Print Port Configure | | | X |
| RASTER OPERATIONS | Raster Memory Write, Absolute | | | X |
| | Raster Memory Write, Relative | | | X |
| | Raster Memory Read Request, Absolute | | | X |
| | Raster Memory Read Request, Relative | | | X |
| | Raster Memory Read Response | | | X |
| | DMA Write, Absolute | | | X |
| | DMA Write, Relative | | | X |
| | DMA Read, Absolute | | | X |
| | DMA Read, Relative | | | X |
| REPORT | Print Buffer Request | | | X |
| | Auxilliary Memory Checksum | | | X |
| | ST3220 System Status Request | | | X |
| | I-R System Status Request | | | X |
| REPORTS | Device Status | X | X | X |
| | Printer Status | X | | X |
| | Cursor Position Report | X | X | X |
| | Device Attributes | X | X | X |
| | Identify Terminal | X | X | X |
| | Request Terminal Params | X | X | |
| RESET MODE (ANSI) | Unlock Keyboard | X | X | X |
| | Replace Mode | X | | |
| | Local Echo On | X | X | |
| | CR Only | X | X | X |
| RESET MODE (DEC) | Cursor ANSI | X | X | X |
| | VT52 Mode | X | X | X |
| | 80 Columns | X | X | |
| | Jump Scroll | X | X | X |
| | Normal Video (Screen) | X | X | X |
| | Origin Reset | X | X | X |
| | Auto Wrap Off | X | X | X |
| | Auto Repeat Off | X | X | X |
| | No Print Formfeed | X | X | X |
| | Text Cursor Off | X | | X |
| RESET MODE (DeeCO) | Screen Saver Off | | | X |
| | Cursor Underline Style | | | X |
| | Cursor Blink Off | | | X |
| | Print Input Disable | | | X |
| | Print Input No Buffer | | | X |
| | Clipping Disable | | | X |
| | Graph String No-Wrap | | | X |
| | Graph Cursor Off | | | X |
| | Graph Cursor X Type | | | X |

## COMMAND COMPARISON CHART (con't.)

(X = Featured)

| COMMAND | | VT-220 | VT-100 | DeeCO ST3220 |
|---|---|---|---|---|
| RESET | Reset To Initial State | X | X | X |
| | Soft Reset | X | | |
| SCROLL | Scroll Up | X | | X |
| | Scroll Down | X | | X |
| | Scroll Margins | X | | X |
| SET MODE (ANSI) | Lock Keyboard | X | X | X |
| | Insert Mode | X | | |
| | Local Echo Off | X | X | |
| | CR & LF (New Line) | X | X | X |
| SET MODE (DEC) | Cursor Application | X | X | X |
| | ANSI Mode | X | X | X |
| | 132 Columns | X | X | |
| | Smooth Scroll | X | X | |
| | Reverse Video (Screen) | X | X | X |
| | Origin Set | X | X | |
| | Auto Wrap On | X | X | X |
| | Auto Repeat On | X | X | X |
| | Print Formfeed | X | | X |
| | Print Extend - Screen | X | | X |
| | Text Cursor On | X | | X |
| SET MODE (DeeCO) | Screen Save On | | | X |
| | Cursor Block Style | | | X |
| | Cursor Blink On | | | X |
| | Print Input Enable | | | X |
| | Print Input Buffer | | | X |
| | Clipping Enable | | | X |
| | Graph String Control On | | | X |
| | Graph String Wrap | | | X |
| | Graph Cursor On | | | X |
| | Graph Cursor Box Type | | | X |
| TABS | Horizontal Tab Set | X | X | X |
| | Horizontal Tab Clear | X | X | X |
| | Clear All Tabs | X | X | X |
| USER DEFINED KEYS | Define Keys | X | | |
| | Key Status | X | | |
| VIEWPORT CONTROL | Define Viewport | | | X |
| | Locate Viewport | | | X |
| VT52 ESCAPE SEQUENCES | Cursor Up | | | X |
| | Cursor Down | | | X |
| | Cursor Right | | | X |
| | Cursor Left | | | X |
| | Cursor Home | | | X |
| | Reverse Line Feed | | | X |
| | Erase To End Of Screen | | | X |
| | Erase To End Of Line | | | X |
| | Direct Cursor Address | | | X |
| | Identify Terminal | | | X |
| | Alternate Keypad | | | X |
| | Numeric Keypad | | | X |
| | ANSI Mode | | | X |
| | Auto-Print On | | | X |
| | Auto-Print Off | | | X |
| | Print Control On | | | X |
| | Print Control Off | | | X |
| | Print Screen | | | X |
| | Print Cursor Line | | | X |
| | Enter Graphics | | | X |
| | Exit Graphics | | | X |

# APPENDIX F
# COMMAND SUMMARY

## Supported VT100/VT220 Commands:

| | Command | Command Sequence |
|---|---|---|
| SCROLLING | Scroll Up | CSI[1] Pn S |
| | Scroll Down | CSI Pn T |
| | Scroll Margins | CSI Pt ; Pb r |
| CURSOR POSITIONING | Cursor Up | CSI Pn A |
| | Cursor Down | CSI Pn B |
| | Cursor Right | CSI Pn C |
| | Cursor Left | CSI Pn D |
| | Cursor Position | CSI Pl ; Pc H |
| | Cursor Home | CSI H |
| | Horiz. & Vert. Position | CSI Pl ; Pc f |
| | Index | ESC D |
| | Reverse Index | ESC M |
| | Next Line | ESC E |
| CHARACTER ATTRIBUTES | All Attributes Off | CSI m or CSI 0 m |
| | Reverse Video On | CSI 7 m |
| | Reverse Video Off | CSI 27 m |
| | Multiple Attr. On/Off | CSI Ps1 ; Ps2 ; .. m |
| EDITING | Erase Line From Cursor | CSI K or CSI 0 K |
| | Erase Line To Cursor | CSI 1 K |
| | Erase Line | CSI 2 K |
| | Erase Screen From Cursor | CSI J or CSI 0 J |
| | Erase Screen To Cursor | CSI 1 J |
| | Erase Screen | CSI 2 J |
| PRINTING | Auto Print On | CSI ? 5 i |
| | Auto Print Off | CSI ? 4 i |
| | Print Control On | CSI 5 i |
| | Print Control Off | CSI 4 i |
| | Print Cursor Line | CSI ? 1 i |
| | Print Screen | CSI i or CSI 0 i |
| REPORTS | Device Status | CSI 5 n |
| | Printer Status | CSI ? 15 n |
| | Cursor Pos. Report | CSI 6 n |
| | Identify Terminal | ESC Z |
| RESET | Reset To Initial State | ESC c |

[1] ASCII EQUIVALENT: ESC [

| | Command | Command Sequence |
|---|---|---|
| TABS | Horiz. Tab Set | ESC H |
| | Horiz. Tab Clear | CSI g or CSI 0 g |
| | Clear All Tabs | CSI 3 g |
| SET MODE (ANSI) | Lock Keyboard | CSI 2 h |
| | CR & LF (New Line) | CSI 20 h |
| SET MODE (DEC) | ANSI Mode | CSI ? 2 h |
| | Reverse Video (Screen) | CSI ? 5 h |
| | Auto Wrap On | CSI ? 7 h |
| | Auto Repeat On | CSI ? 8 h |
| | Print Formfeed | CSI ? 18 h |
| | Print Extent = Screen | CSI ? 19 h |
| | Text Cursor On | CSI ? 25 h |
| RESET MODE (ANSI) | Unlock Keyboard | CSI 2 l |
| | CR Only | CSI 20 |
| RESET MODE (DEC) | VT52 Mode | CSI ? 2 l |
| | Normal Video (Screen) | CSI ? 5 l |
| | Auto Wrap Off | CSI ? 7 l |
| | Auto Repeat Off | CSI ? 8 l |
| | No Print Formfeed | CSI ? 18 l |
| | Print Extent = Scroll | CSI ? 19 l |
| | Text Cursor Off | CSI ? 25 l |
| EIGHT BIT C1 CONTROL CODES | Transmit 7-bit Equiv. | ESC <space> F |
| | Transmit 8-bit Control | ESC <space> G |

## VT52 ESCAPE SEQUENCES

Note: These commands are only accessible when in VT52 mode. VT52 mode is entered by the "VT52 Mode" command of the DEC reset mode commands. When in VT52 mode, these are the only commands available. To use the other commands, it is necessary to exit VT52 mode by issuing the "ANSI Mode" command (below).

| | |
|---|---|
| Cursor Up | ESC A |
| Cursor Down | ESC B |
| Cursor Right | ESC C |
| Cursor Left | ESC D |
| Enter Graphics | ESC F |
| Exit Graphics | ESC G |
| Cursor Home | ESC H |
| Reverse Line Feed | ESC I |
| Erase To End Of Screen | ESC J |
| Erase To End Of Line | ESC K |
| Direct Cursor Address | ESC Y Line Column |
| Identify Terminal | ESC Z |
| Alternate Keypad | ESC = |
| Numeric Keypad | ESC > |
| ANSI Mode | ESC < |
| Auto-print On | ESC ^ |
| Auto-print Off | ESC _ |
| Print Control On | ESC W |
| Print Control Off | ESC X |
| Print Screen | ESC ] |
| Print Cursor Line | ESC V |

### 4.3.2 DeeCO Extension to VT100/VT220 Commands:

| | Command | Command Sequence |
|---|---|---|
| SET/RESET MODE (DeeCO) | Screen Saver On | CSI < 1 h |
| | Screen Saver Off | CSI < 1 l |
| | Cursor Block Style | CSI < 2 h |
| | Cursor Underline Style | CSI < 2 l |
| | Print Input Enable | CSI < 4 h |
| | Print Input Disable | CSI < 4 l |
| | Print Input Buffer | CSI < 5 h |
| | Print Input No Buffer | CSI < 5 l |
| | Clipping Enable | CSI < 6 h |
| | Clipping Disable | CSI < 6 l |
| | Graph String Control On | CSI < 7 h |
| | Graph String Control Off | CSI < 7 l |
| | Graph String Wrap | CSI < 8 h |
| | Graph String No-Wrap | CSI < 8 l |
| | Graph Cursor On | CSI < 9 h |
| | Graph Cursor Off | CSI < 9 l |
| | Graph Cursor Box Type | CSI < 10 h |
| | Graph Cursor X Type | CSI < 10 l |
| REPORT | Print Buffer Request | CSI < 1 n |
| | Print Buffer Response | DCS[2] b <a..a> ST[3] |
| | Auxiliary Memory Checksum Request | CSI < 2 n |
| | Auxiliary Memory Checksum Response | CSI < 2; Pr; Pc n |
| | ST3220 System Status Request | CSI < 3 n |
| | ST3220 System Status Respons | CSI < 3; Pv; Pr; Pm; Pi; Ps; Pa n |
| PRINTING | Print Port Configure | CSI < Pb; Pd; Ps; Pp c |
| ALPHA CONTROL | Alpha Character Size | CSI < Ps s |
| | Character Set Select | CSI < Ps S |
| | Alpha Write Mode | CSI < Ps M |
| CHARACTER ATTRIBUTES | Italic Attribute On: | CSI < 9 m |
| | Italic Attribute Off: | CSI < 29 m |
| GRAPH WINDOW COMMAND CONTROL | Define Window Area | CSI < Pyl; Pxl; Pyu; Pxr w |
| | Define Window Origin | CSI < Pyo; Pxo o |
| | Define Window Type | CSI < Pt W |
| | Erase Graphic Window | CSI < e (and:DCS O G ST or DCS G ST) |
| VIEWPORT CONTROL | Define Viewport | CSI < Pv; Ps v |
| | Locate Viewport | CSI < Pv; Pt; Py; Px p |
| DATA/DISPLAY LISTS | Open Display/Data List | CSI < Pid; Pt; Pa d |
| | Close List | Control-\ (1CH) |
| | Execute Display List | CSI < Pid; Px; Py x |
| | Append Data List | DCS Pid a <h..h> ST |
| | Read Data List | CSI < Pid r |
| | List Report | DCS Pid; Ps; Pt r <h..h> ST |
| | Clear All Lists | CSI < C |

[2] ASCII EQUIVALENT: ESC P
[3] ASCII EQUIVALENT: ESC \

| | Command | Command Sequence |
|---|---|---|
| AUXILIARY MEMORY | Auxiliary Memory Type | CSI < Pt m |
| | Compute/Store Checksum | CSI < a |
| CHARACTER DEFINITION | Define Alternate Character | DCS 33 G Pc; P10; P11; ... P17 ST |
| OTHER | Screen Saver Timeout | CSI < Pn t |
| GRAPHIC OBJECT | Clear Graphic Window | DCS 0 ST or DCS G ST |
| | Set Graphic Cursor | DCS 1 G Py; Px ST |
| | Vector Draw, Absolute | DCS 2 G Py1; Px1; Py2; Px2 ST |
| | Vector Draw, Relative | DCS 3 G Py2; Px2 ST |
| | Circle Draw, Absolute | DCS 4 G Pr; Pyc; Pxc ST |
| | Circle Draw, Relative | DCS 5 G Pr ST |
| | Arc Draw, Absolute | DCS 6 G Pa0; Pa1; Pr; Pyc; Pxc ST |
| | Arc Draw, Relative | DCS 7 G Pa0; Pa1; Pr ST |
| | Block Fill Draw, Absolute | DCS 8 G Py1; Px1; Py2; Px2 ST |
| | Block Fill Draw, Relative | DCS 9 G Py2; Px2 ST |
| | Concave Polygon Fill, Absolute | DCS 10 G Py; Px ST |
| | Concave Polygon Fill, Relative | DCS 11 G ST |
| | Convex Polygon Fill, Absolute | DCS 12 G Py; Px ST |
| | Convex Polygon Fill, Relative | DCS 13 G ST |
| | Graphic String Draw, Absolute | DCS 14 G Py; Px A <....> ST |
| | Graphic String Draw, Relative | DCS 15 G A <....> ST |
| GRAPHIC ATTRIBUTES | Character Attribute Select | DCS 16 G Ps; Pr; Pi; Pu; Pc ST |
| | String Attribute Select | DCS 17 G Pd; Pp ST |
| | Line Style Attribute Select | DCS 18 G Ps ST |
| | Fill Style Attribute Select | DCS 19 G Ps ST |
| | Graph Write Mode Select | DCS 20 G Ps ST |
| RASTER OPERATIONS | Raster Memory Write, Absolute | DCS 21 G Py; Px H <h..> ST |
| | Raster Memory Write, Relative | DCS 22 G H <h..> ST |
| | Raster Memory Read Request, Absolute | DCS 23 G Py; Px; Pc ST |
| | Raster Memory Read Request, Relative | DCS 24 G Pc ST |
| | Raster Memory Read Response | DCS 25 G <h..> ST |
| AREA FLASH | Flash Area, Define | DCS 30 G Pi; Pl; Pyl; Pxl; Pyu; Pxr ST |
| | Flash Area, Start | DCS 31 G Pi ST |
| | Flash Area, Stop | DCS 32 G Pi ST |
| FAST VECTOR | Enter Fast Vector Mode | DCS 34 G ST <vector data ..> |

# APPENDIX G
# RETAINED GRAPHIC FRAME STORAGE

"Display lists" are sequences of commands stored in terminal memory. "Data lists" are sequences of data bytes stored in terminal memory, whose meaning is defined by the user/host. The terminal provides EEPROM (non-volatile) or RAM (volatile) memory for these lists. The list memory is 16K x 8. The terminal determines on power-up how big the list memory is (if it is loaded). The first 32 bytes of list memory are reserved for system control. If present and properly structured, the first byte contains the checksum of list memory. Commands exist to verify this checksum. The next four bytes contain the sequence 49H, 4EH, 49H, 54H. This sequence, if present, signals a valid list memory. The integrity of the list memory is verified by the checksum. The last 4K of the list memory may be used for user-defined characters, if desired.

A "list" is comprised of a list header and a list body. The header is 8 bytes long, the body can be of any length up to a size compatible with the size of list memory. The list headers form a linked list structure starting at the low address bound + 20H of the list memory. The header contains 4 fields:

| Field Name | Function |
|---|---|
| ID | 12-bit id number + 4 bit flags |
| Body Pointer | 16-bit pointer to the body of the list. |
| Body Size | 16-bit byte count of the body. |
| Next Header Pointer | Link pointer to the next list header. |

The ID is used to access the list. This number is specified in commands to uniquely identify the list. The ID of the end-of-header-list contains the value FFFFH. The upper four bits of the ID field contain four 1-bit flags. The first of these flags (bit number 12), is used to contain the list type, 0=display list, 1=data list. The other three bit flags are unused at present.

The Body Pointer contains the absolute address of the body of the list. This address may be anywhere in list memory. Typically the body of any list which is static (never redefined) immediately follows the header. Lists which are re-written often have their bodies elsewhere in list memory so the size may vary.

The Body Size specifies the size of the body of the list in bytes.

The Next Header Pointer is used to contain the address of the next header in list memory.

The number of lists is restricted by the size of list memory. Each list is 8 + (body size) bytes long, the sum of the sizes of all lists must be less than or equal to the list memory size - 32 (for list memory control structure). An additional 2K must be subtracted if user-defined characters are used.

Display list may be "executed" by the terminal. The host specifies the list to be executed by its ID number. A list must be display list type, not a data list type, to be executed.

The data list type is provided to allow the host to take advantage of the terminal's non-volatile memory. Data which the host wants to store in such memory may be stored in a data list. Commands exist for incrementally adding data to the list in cases where all the data is not available at once.

Any list type, data or display may be read by the host. The terminal will, upon direction by the host, transmit the body encoded in hex-ASCII of a list to the host.

## ALTERNATE CHARACTER SET

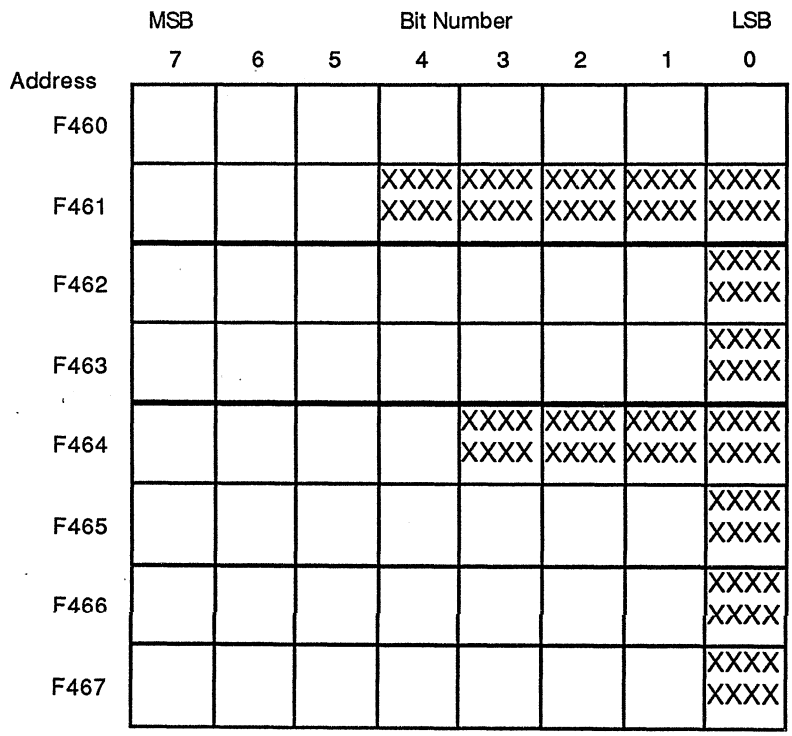
The alternate character set refers to the set comprised of characters defined by the host. These character images are stored in auxiliary memory in the last 4K bytes.

A character image consists of n bytes which define an 8xn pixel array. For 640x400 displays, n is 16. When the character is drawn, this image is transferred to raster memory at the designated point. The character is accessed by 1) invoking the alternate character set, and 2) identifying the character by its 8-bit code (such as an ASCII code).

As an example, consider how the character "F" would be stored in the alternate character set. First of all, the address in auxiliary memory where the character is physically stored is defined by the character's code:

storage address = F000 + (Code * 16)

If the code were 46H, the ASCII code for "F", the first address of the character image would be at F460. The first 8 bytes which define the character would look like this in auxiliary memory:




Where a blank bit cell denotes a "0" and a cell filled with "X"'s denotes a "1".

In hex the 8 bytes are: 0, 1F, 1, 1, F, 1, 1, 1.

# APPENDIX H
# RS-232C

## BACKGROUND

Although it is commonly referred to as a standard for computer communications, RS-232C was in fact defined and developed for telecommunications systems. Consequently, because its application to computer systems is not quite compatible with its original purpose, there are many different interpretations and implementations used today. Users must be very careful to understand exactly how RS-232C has been implemented in the equipment they are using. There is no guarantee whatever that two systems, when connected together by RS-232C, can communicate.

In its original conception, the RS-232C was defined as the interface between a remote terminal and a modem connected to a telecommunications channel (see Figure 1 below).

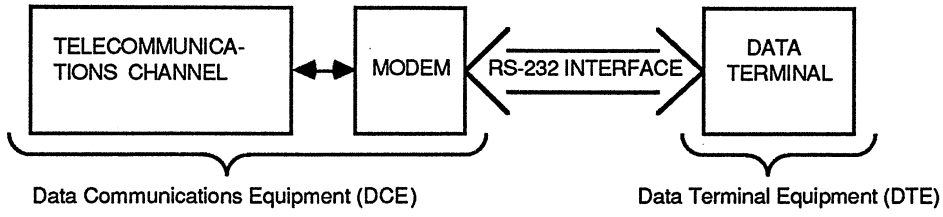



**Figure 1: BLOCK DIAGRAM OF PART OF A TELECOMMUNICATIONS SYSTEM SHOWING FUNCTIONAL POSITION OF RS-232C INTERFACE**

Since the cable was expected to be a straight pin-to-pin connection, two types of connectors were defined and known respectively, as DTE (Data Terminal Equipment) and DCE (Data Communications Equipment). Of the 25 lines defined in the official standard and shown in Figure 3, only 9 at the very most, are ever used on computer interconnects. They are identified in Figure 2, below.

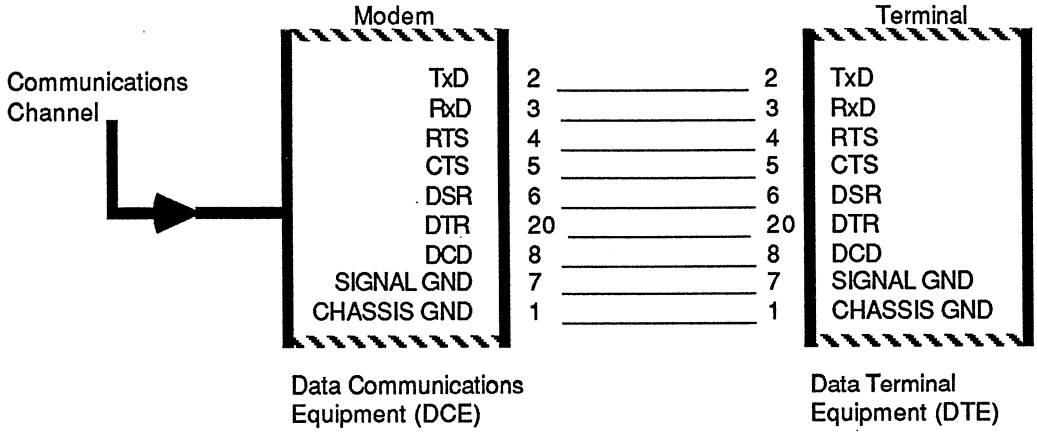



**Figure 2: RS-232C INTERFACE DEFINITION IN TELECOMMUNICATIONS SHOWING LIMITED SIGNAL SET**

## COMPUTER USE

There are some basic differences between the original telecommunications hook-up (i.e. terminal to modem), and the computer to computer, or computer to terminal situations.

In the case of two computers or terminals, using a two-way data channel (full-duplex), both devices can be assumed to be active all the time, and none of the handshaking signals required for establishing communications links in a single-channel switched telecommunications network are required. In fact, for the computers and terminals, the only handshaking required is to indicate busy/not busy. This can be done with only two lines, one driven by each device.

This protocol is typically implemented using the RTS/CTS or the DTR/DSR pairs of lines.

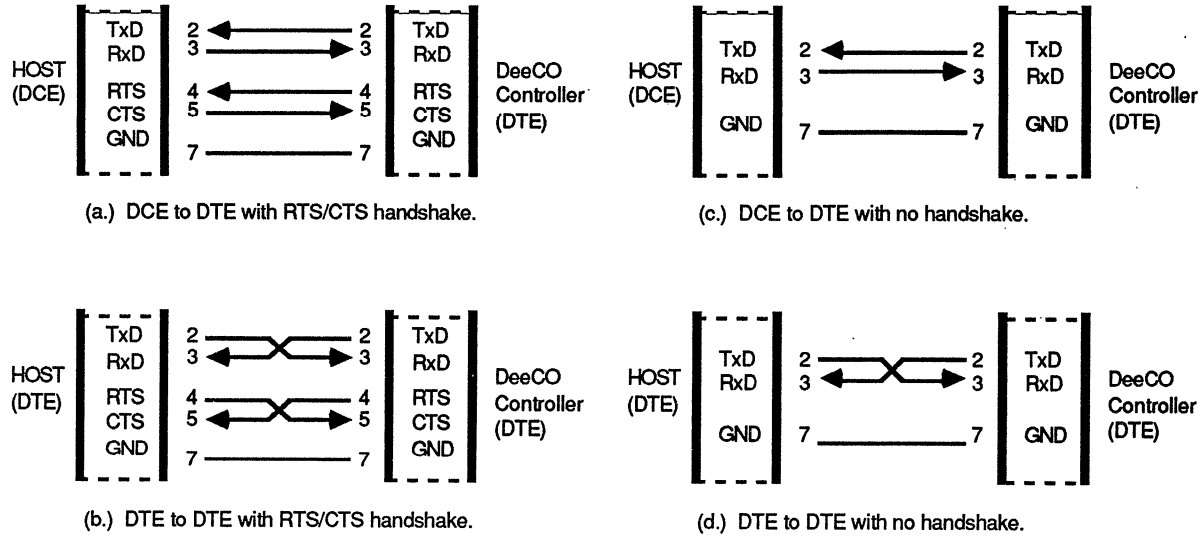
The descriptions DTE (Data Terminal Equipment) and DCE (Data Communications Equipment) for the connectors is a carry over from the original specifications. The primary difference between the two in a computer to terminal situation is that they define different pin-outs (e.g. received and transmitted data pins are swapped). It is important to check the style of connectors before connecting two pieces of equipment together. DeeCO terminals always use the DTE style of connection.

## DEECO IMPLEMENTATION – FULL DUPLEX

Lucas Deeco Corporation terminal systems implement the RTS/CTS style of the interface. Recommended wiring connections for some different situations are shown in Figure 4. Note that the pin numbers shown here are those of the standard RS-232C D-Subminiature 25-pin connector. On the DeeCO M4ST product, pins 12, 13, 24 and 25 have been used to implement RS 485 multidrop, as described in Section 5.

TxD and RxD are the lines assigned for transmission and reception of data. RTS will be asserted by the terminal when the input data buffer is 80% full. This should cause the host to stop sending further data. When the buffer is reduced to 20% of its capacity, the RTS signal will be de-asserted so that communications from the host can resume.

CTS is an input to the terminal which is used to control transmission from the terminal to the host. While the CTS line is asserted the data transmission will be inhibited.




(a.) DCE to DTE with RTS/CTS handshake.

(c.) DCE to DTE with no handshake.

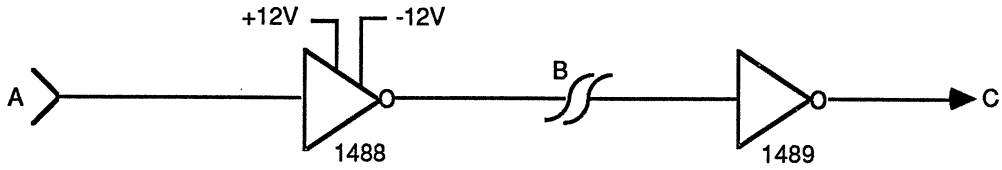
(b.) DTE to DTE with RTS/CTS handshake.

(d.) DTE to DTE with no handshake.

**Figure 4: VARIOUS POSSIBLE HOOK-UPS BETWEEN A HOST SYSTEM AND THE DEECO TERMINAL**

For the purpose of this discussion "asserted" is defined as the state of the signal which will produce a logic "I" (+5V) at the output of the RS-232C line receiver. "De-asserted" is the opposite condition (see Figure 5).

A parallel "busy" protocol is implemented in software. At the same time as the terminal sends the "busy" indication on RTS, it also transmits an XOFF message to the host which is an alternative indication for busy. The return to a ready condition is signaled by the message XON.




| | A | | B | | C | |
|---|---|---|---|---|---|---|
| Signal State | Logic | Voltage | Voltage | Logic | Logic | Voltage |
| Asserted | "1" | +5V | -12V | Mark | "1" | +5V |
| De-asserted | "0" | 0V | +12V | Space | "0" | 0V |

**Figure 5: TYPICAL RS-232C TRANSMISSION LINE SCHEMATIC AND TABLE SHOWING THE CONDITIONS AT ITS THREE SECTIONS DURING BOTH ALLOWABLE STATES**

## INCOMPATIBLE SYSTEMS

Although most computer systems can be set up to operate in the manner just described, there are many systems which cannot. Their operation more closely resembles that of the original handshake signal definitions and may not be accessible to changes. In those cases the best approach is to connect up only the data lines and, of course, signal ground.

In this case, the software XON/XOFF protocol should be used in conjunction with a lower baud rate (e.g. 9600 or lower). Since the software protocol is not as effective in halting transmission of data, the use of a lower baud rate gives the sending system more time to react.

# APPENDIX I
# KEYBOARD

The proper keyboard to be connected to the interface connector is a 101/102 enhanced AT style keyboard. Scan code set 2 will be used.

The keyboard may be used in either a mapped or unmapped mode. In the unmapped mode the key scan codes are sent directly to the host uninterpreted by the ST3220. When mapped, the codes sent to the host are ASCII where possible and where ASCII is not possible (such as function keys) a unique ANSI 3.64 sequence will be sent to identify the key. Included in the sequence will be information about the current state of the shift keys SHIFT, CTRL, and ALT.

ASCII-possible keys will send the ASCII code of the character indicated by the key's legend. Typing CTRL in conjunction with the key will produce the ASCII code AND'ed with 9F hex to produce an ASCII control code. Using the ALT key will result in the ASCII code OR'd with 80H to produce full 8-bit codes.

The non-ASCII keys SCROLL LOCK and BREAK may be used to affect the host communication. SCROLL LOCK will act as a toggle to send an XOFF or XON. BREAK will issue a 'break' on the RS232 communication line.

The key SYSREQ (PRINT SCREEN) is a toggle which will select or de-select the on-screen set-up mode.

Other non-ASCII keys will send the following ANSI 3.4 sequences:

| Key Legend | ANSI Sequence |
|---|---|
| up arrow | CSI A |
| dn arrow | CSI B |
| lf arrow | CSI C |
| rt arrow | CSI D |
| | |
| F1 | CSI = Ps a |
| F2 | CSI = Ps b |
| F3 | CSI = Ps cj |
| F4 | CSI = Ps d |
| F5 | CSI = Ps e |
| F6 | CSI = Ps f |
| F7 | CSI = Ps g |
| F8 | CSI = Ps h |
| F9 | CSI = Ps i |
| F10 | CSI = Ps j |
| F11 | CSI = Ps k |
| F12 | CSI = Ps l |
| | |
| Insert | CSI = Ps p |
| Home | CSI = Ps q |
| Page Up | CSI = Ps r |
| Delete | CSI = Ps s |
| End | CSI = Ps t |
| Page Down | CSI = Ps u |

The key pad will send the ASCII code corresponding to the character on the key's legend when Num Lock is on.

When Num Lock is off the following will be sent:

| **Key Pad (Num Lock Off)** | |
|---|---|
| / | CSI = Ps K |
| * | CSI = Ps L |
| - | CSI = Ps M |
| . | CSI = Ps N |
| Enter | CSI = Ps O |
| 9 | CSI = Ps J |
| 8 | CSI = Ps I |
| 7 | CSI = Ps H |
| 6 | CSI = Ps G |
| 5 | CSI = Ps F |
| 4 | CSI = Ps E |
| 3 | CSI = Ps D |
| 2 | CSI = Ps C |
| 1 | CSI = Ps B |
| 0 | CSI = Ps A |

The parameter Ps in the above sequences is a bit-encoded number describing the current state of the shift keys SHIFT, CTRL, and ALT. If Ps is missing then it is assumed to be zero (0). The relation between Ps and the shift keys is as follows:

| **Ps** | **SHIFT** | **CTRL** | **ALT** |
|---|---|---|---|
| 0 | no | no | no |
| 1 | yes | no | no |
| 2 | no | yes | no |
| 3 | yes | yes | no |
| 4 | no | no | yes |
| 5 | yes | no | yes |
| 6 | no | yes | yes |
| 7 | yes | yes | yes |

# APPENDIX J
# POWER-UP MESSAGES

The ST3220 will perform a variety of tests upon power-up and report the results via messages on the screen. These messages will remain for approximately two seconds then will disappear.

The ST3220 will report its self-test results on the first line of the display. The message:

SELF TEST OK

will indicate that all tests passed. The message:

SELF TEST FAIL: XX

will indicate that one or more of the self test diagnostics failed. Which tests fail will be indicated by the 2 digit hexadecimal number "XX'. This is a bit encoded 8-bit number where each bit indicates a particular test. The test/bit relationship is as follows:

| Bit Number | Test |
|---|---|
| 0 | ROM checksum |
| 1 | System RAM |
| 2 | IR subsystem |
| 3 - 7 | Undefined |

The ST3220 will also display a message indicating where it obtained its default operating modes. The defaults may be retrieved from non-volatile memory or be specified by jumpers. The following messages may occur:

CONFIGURATION BY STATE MEMORY

This message will appear when the default operating modes have been obtained by the values stored in non-volatile state memory. It indicates that the state information has been successfully retrieved without error.

STATE MEMORY INVALID: ERROR

This message indicates that an attempt to use the non-volatile state memory values failed. The data in the non-volatile memory was most likely lost. The defaults used will be determined by factory defaults and switch settings.

STATE MEMORY CHANGES: STATE DEFINITION

This message indicates that an attempt to use the non-volatile state memory values failed. The data in the non-volatile memory is intact but the system has determined that it is insufficient. The cause of this message is usually a new software revision that has changed the definition of "STATE": either added or subtracted information from it. The defaults used will be determined by factory defaults and switch settings.

STATE MEMORY CHANGES: SWITCH PRIORITY

This message indicates that an attempt to use the non-volatile state memory values failed. The data in the non-volatile memory is intact but the configuration jumpers have changed since the state was stored. The new switch settings will be given priority. The defaults used will be determined by factory defaults and switch settings.

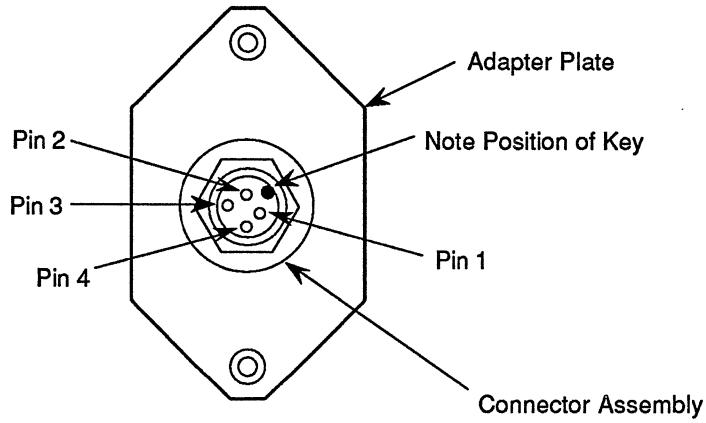
# APPENDIX K
# ST3225 SPECIFICATIONS

The ST3225 is a DC powered version on the ST3220. The software and computer hardware of the ST3225 is 100% compatible with the ST3220. The difference is in the Power to the ST3225 which consists of +5VDC and +12VDC rather than the 120VAC on the ST3220.

Current requirements for the ST3225 are:

| | |
|---|---|
| +5VDC +5% -0% at | 1.1A Typical |
| +12VDC +5% -0% at | 1.2A Typical |

The power connector for the ST3225 is shown below.




**Front View of Connector**

The ST3225 comes with a mating cable for the power connector that includes a molded 12 foot long cable. The user must wire the end on this cable to their +5 and +12VDC power supply as follows:

| Pin | Wire Color | Connect to |
|---|---|---|
| 1 | Brown | Chassis Ground |
| 2 | White | +5VDC |
| 3 | Blue | +12VDC |
| 4 | Black | Signal Ground |

# APPENDIX L
# INSTALLATION PROCEDURES FOR CONDUIT COVERS

The list of parts for the Conduit Cover kit is:

| Qty | Description |
|---|---|
| 1 | Cast Aluminum Power Cover |
| 1 | Rubber Gasket for Power Cover |
| 4 | 4-40 x 1.00" Allen Head Screws |
| 1 | Power Cord ** |
| | |
| 1 | Cast Aluminum Communications Cover |
| 1 | Rubber Gasket for Power Cover |
| 6 | 4-40 x 1.00" Allen Head Screws |
| 1 | 25 pin D-Sub, Solder tab Connector |
| 1 | 9 pin D-Sub, Solder tab Connector |
| 4 | 4-40 x 2.5" pan head screws for D-subs |

** The power cord has been left intact for shipping to allow the user to power the terminal for check out without having to assemble and wire with the conduit cover first.

**Installing the Power Conduit Cover:**

1. Place the gasket over the four mounting studs on the back of the terminal.
2. Cut the male end of the power cord off and feed that end through the threaded hole in the cover.
3. Insert the female end of the power cord into the power socket on the back of the terminal.
4. Place the conduit cover over the same studs.
5. Screw in place with the four 4-40 x 1.25" Allen Head Screws.
6. Insert a threaded 1/2" conduit connector to the conduit cover (1/2" to 3/4" adapter may be used if desired).
7. Strip the loose end of the power cord and connect to an appropriate 115VAC screw terminal outlet.

**Installing the Communications Conduit Cover:**

1. Place the gasket over the six mounting studs on the back of the terminal.
2. Insert a threaded 1/2" conduit connector to the conduit cover (1/2" to 3/4" adapter can be used if desired).
3. Feed the cable through the cover from the outside to the inside.
4. Solder the appropriate wires to the 25 pin Host Communications connector using Appendix B in the manual as a signal assignment reference. Secure the connector in place with two 4-40 screws.
5. Repeat step 4 for the printer connector if a printer is to be used.
6. Place the Conduit cover over the mounting studs and secure with the six 4-40 x 1.25" Allen Head screws.
7. Attach the other end of the host communication (printer port) cable(s) to their appropriate host (printer) connection.

# APPENDIX M
# DOCUMENTATION, ORDERING AND CUSTOMER SUPPORT

## DOCUMENTATION

At this writing, all available documentation on the SealTouch Terminal (Model ST3220) is contained within this manual.

## ORDERING

The following information is required prior to calling Lucas Deeco to place an order:

- Company Name
- Complete Billing Address
- Ship To Address
- Purchase Order Number
- Buyer's Name
- Buyer's Telephone Number
- Technical Contact's Name (for product update information)
- Technical Contact's Telephone Number
- Method of Shipment
- Whether order is resale or not (if order placed for California company)
- Product Order
- Quantity of Product Order

Method of Shipment: U.P.S. Blue, unless otherwise requested by customer.

Delivery: 30 to 60 days ARO, unless customer requires delivery sooner. The operations department is consulted prior to quoting expedient delivery dates.

Contact:

Lucas DeecoCorporation
31047 Genstar Road
Hayward, California 94544-7831
FAX: (415) 489-3500
Telephone: (415) 471-4700

## CUSTOMER SUPPORT

Lucas Deeco Corporation places great importance on building and maintaining excellent customer relations. Customer feedback and communication are invaluable tools in providing the best possible products and services. If you have not already done so, kindly take a moment to complete and return the Purchaser's Registration Card provided in the front of this manual. This will assist us in notifying you of any changes, enhancements or updates to the product.

Should you require operational assistance, or have other questions relative to the ST3220, our Customer Service personnel are trained to provide technical assistance by calling, (415) 471-4700, Monday through Friday, from 8:00 A.M. to 12:00 P.M. and from 1:00 P.M. to 5:00 P.M. Pacific Standard Time.

# APPLICATION NOTES

## 1. SEALTOUCH SOFTWARE EXAMPLES

### 1.1. DEFINING BUTTON LOCATIONS AND RESPONSES

Define a button to be located in the lower-left corner of the display. When pressed, the button will send the ASCII characters "HOST" out the serial port to the host computer. The ST3220 will also display the characters "LOCAL" at the current cursor position. The label will be "TEST." The button identification number is 0.

1. To define the size and position of a button, use the following command:

    ESC[>0;1;1;10;10B

Where:

**0** is the button identification number.
**1** is the bottom edge of the button.
**1** is left edge of the button.
**10** is the top edge of the button.
**10** is the right edge of the button.

2. To define the button response, use the following command:

    ESCP~0/host|local:testESC\

Where:

**0** is the button identification number.
**host** is the response sent to the host.
**local** is the response sent locally to the ST3220.
**test** is the label.

### 1.2. USING LOCAL RESPONSES

Local responses are useful for controlling terminal functions without involving the host computer. Local responses may be defined without a host response being present. The command to change a page (to page 1) is:

    ESC[>1P

Where:

**1** is the page number.

In a local response, commands must be given in the following manner:

    ^[[>1P = ESC[>1P

In commands with multiple parameters, parameters are separated by ";" In local responses, ";" must be preceded by a "\" Commands with multiple parameters should be given in the following manner:

    ^[[>0\;1P = ESC[>0;1P

Example:

Define a button to have a local response that will change the current and definition page number to 1 First, define a button location Then, define the button response with:

ESCP~0|^[[>1P:to p.1ESC\

*NOTE:* This local response changes the current button page to 1 Since there is nothing defined on page 1, the screen will be blank after this button is activated.

## 1.3. USING BUTTON PAGES

This example will define three button pages (a main menu page and two action pages) Since a button can have different responses on each page, only button numbers 0 and 1 will be used.

Example:

Button page 0 is the current and definition page at power-up.

| Command: | Description: |
|---|---|
| ESC[>0;1;1;10;10B | Define button 0 location. |
| ESC[>1;10;10;20B | Define button 1 location. |
| ESCP~0\|^[[>1P:to p.1ESC\ | Define button 0 response This is a local response that will change to page 1. |
| ESCP~1\|^[[>2P:to p.2ESC\ | Define button 1 response This is a local response that will change to page 2. |
| ESC[>0;1P | Set current page to 0, and definition page to 1. The following response definitions will be on page 1. |
| ESCP~0/1:one ESC\ | Define button 0 response on page 1. This button will send an ASCII "1" to the host. |
| ESCP~1\|^[[>0P:to p.0ESC\ | Define button 1 response on page 1. This is a local response that will change back to page 0. |
| ESC[>0;2P | Set current page to 0, and definition page to 2. The following response definitions will be on page 2. |
| ESCP~0/2:two ESC\ | Define button 0 response on page 2. This button will send an ASCII "2" to the host. |
| ESCP~1\|^[[>0P:to p.0ESC\ | Define button 1 response on page 2. This is a local response that will change back to page 0. |

## 1.4. USING POP-UP MENUS

This example will define a main menu button and a pop-up page. This pop-up page will have three choices, each choice will send a different string to the host.

Example:

| Command: | Description: |
|---|---|
| ESC[>0;1;1;10;10B | Define button 0 location. |
| ESCP~0\|^[[>1M:p.upESC\ | Define button 0 response. This button will call pop-up page 1. |
| ESC[>0;1M | Set current page to 0 and definition page to be pop-up page 1. |

The following response definitions will be pop-up choices on pop-up page 1.

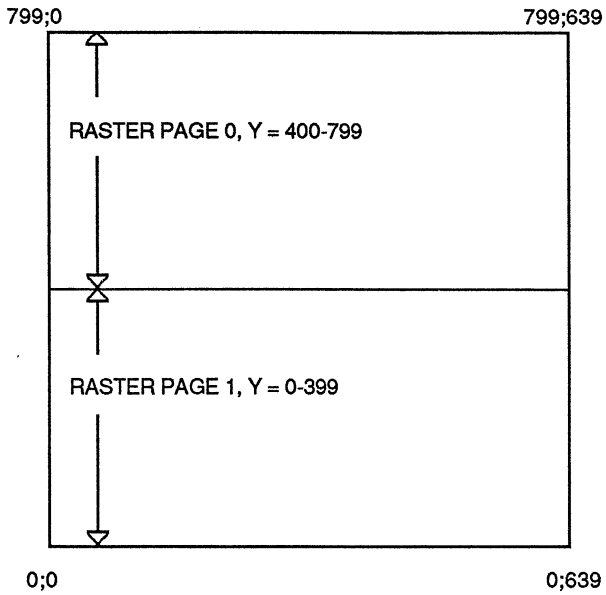
| Command: | Description: |
|---|---|
| ESCP~1/choice 1:one ESC\ | Define pop-up response 1 When activated, this choice will send the ASCII string "choice 1" to the host. |
| ESCP~2/choice 2:two ESC\ | Define pop-up response 2 When activated, this choice will send the ASCII string "choice 2" to the host. |
| ESCP~3/choice 3:three ESC\ | Define pop-up response 3 When activated, this choice will send the ASCII string "choice 3" to the host. |

## 2. RASTER (VIDEO) MEMORY

The M4' s raster (video) memory consists of 640 by 800 memory locations. This memory is divided into two raster pages, Page 0 and Page 1. This basic coordinate system is referred to as the "Global Coordinate System". Coordinates are defined as Y;X. (See FIG.1).

In global coordinates, page 0 is located from 400;0 to 799;639. **Alpha text operations are always performed on raster page 0.** Raster page 1 is located from 0;0 to 399;639. Graphic objects may be drawn on either page, however, if the Alpha page (page 0) is scrolled, anything that is on it ( be it graphics or text), is scrolled along with the page. **Page 0 is the only page that may be scrolled.**

FIG. 1




## 3. GRAPHIC WINDOW

**Graphic objects may be drawn in any part of raster memory.** Graphic objects are drawn using an offset relative to the "graphic window origin" and may be "clipped" by the boundries of a rectangular region known as the " Graphics Window".

*To "clip" an object is to draw only that portion of the object that lies within the " graphic window".* Portions of the object lying outside the window are not drawn, and are therefore "clipped" at the window's boundries.

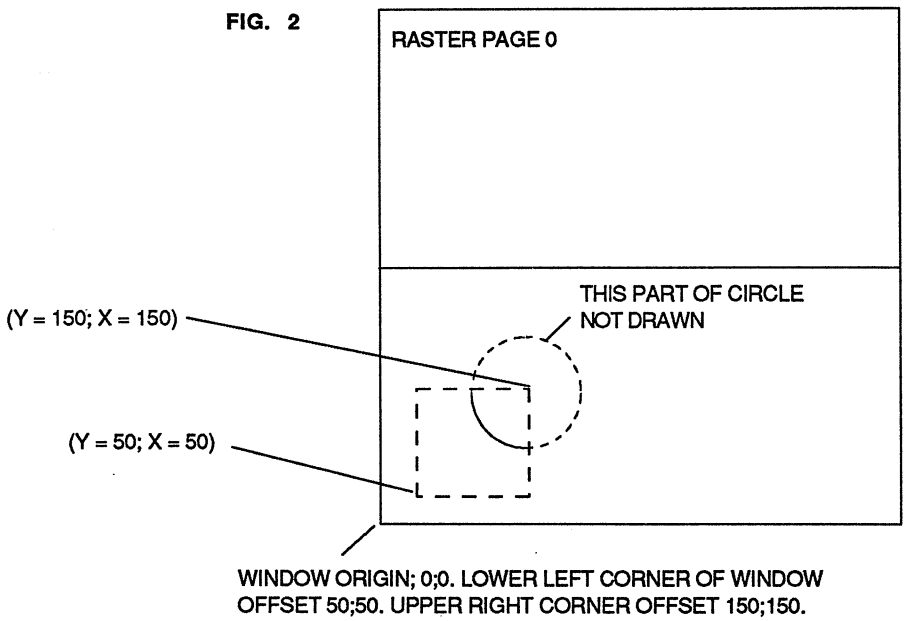
*Clipping is automaticaly enabled by default, and may be disabled by a software command.*

FIG 2 demonstrates clipping. The dotted rectangle represents the boundries of the "Graphic Window". The *origin of the Graphic Window* is set as being *equal to the Global Origin,(0;0 in global coordinates).*

The lower left corner of the graphic window is offset from the graphic window origin by Y=50; X=50. The upper right corner of the Graphic window is offset from the graphic window origin by Y=150; X=150. This creates a Graphic window measuring 100 pixels by 100 pixels.

A circle is drawn with a large enough radius that part of it lies outside the boundries of the "window". *When clipping is enabled, the part of the circle lying outside the window will not be drawn in raster memory.*

FIG. 2


RASTER PAGE 0
THIS PART OF CIRCLE
NOT DRAWN
(Y = 150; X = 150)
(Y = 50; X = 50)
WINDOW ORIGIN; 0;0. LOWER LEFT CORNER OF WINDOW
OFFSET 50;50. UPPER RIGHT CORNER OFFSET 150;150.

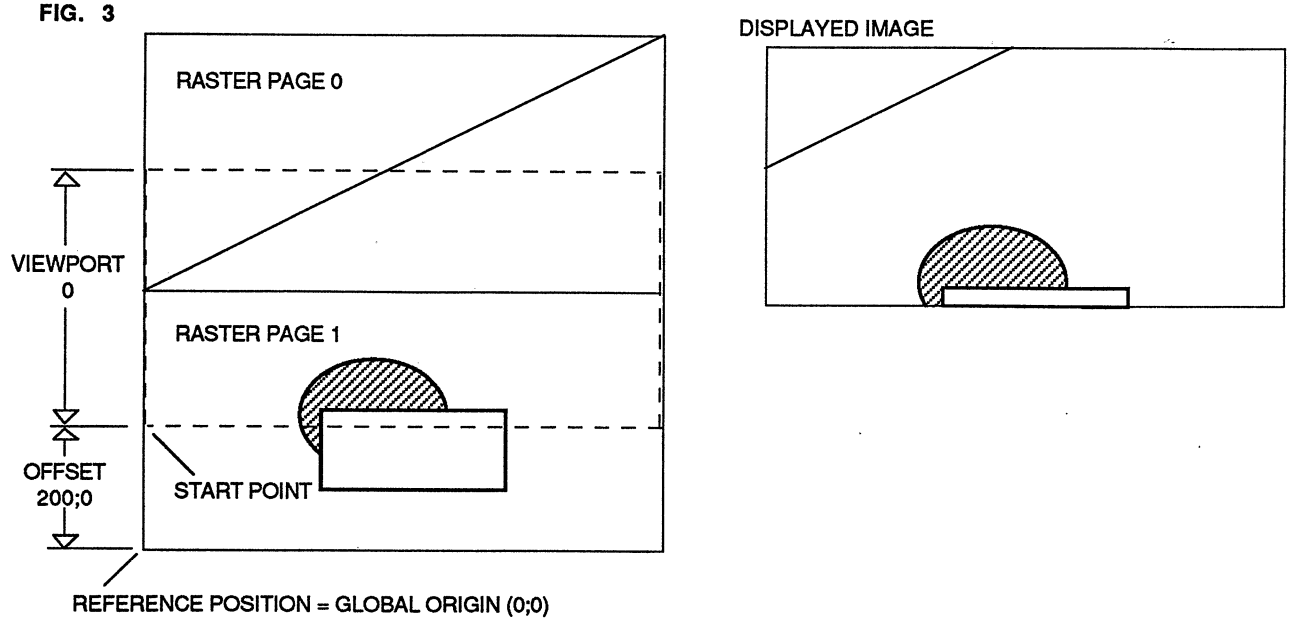
## 4. VIEWPORTS

***"Viewports" do nothing more than specify that portion of raster memory that will be displayed.*** There are two viewports available. These are named Viewport 0 and Viewport 1. **Do not confuse these "viewports" with raster pages 0 and 1.**

Viewport 0 is the background viewport, and viewport 1 is the foreground or overlapping viewport.
The width of a viewport is always the width of the screen.
The size of a viewport is it's height in pixels.
The size of viewport 0 is always 400.
The size of viewport 1 is user programmable.

FIG. 3, 4, and 5 demonstrate viewport functions. In FIG. 3, a diagonal line is drawn in raster page 0 from the lower left corner to the upper right corner. A circle and a rectangle are drawn on raster page 1. The "Start-Point" (lower left corner), of viewport 0 has been defined as being 200;0 (in global coordinates). ***Viewport 0 will cause the top half of raster page 1 and the bottom half of raster page 0 to be displayed.***

FIG. 3




In FIG 4, a diagonal line is drawn on raster page 0 from the lower left corner to the upper right corner. A circle is drawn on raster page 1. The start point of viewport 0 has been defined as being equal to the lower left corner of raster page 0, (400;0 in global coordinates). The start of viewport 1 has been defined as being equal to the lower left corner of raster page 1, (0;0 in global coordinates). *Because the height of viewport 1 is 0, only the raster memory specified by viewport 0 is displayed.*

FIG. 4

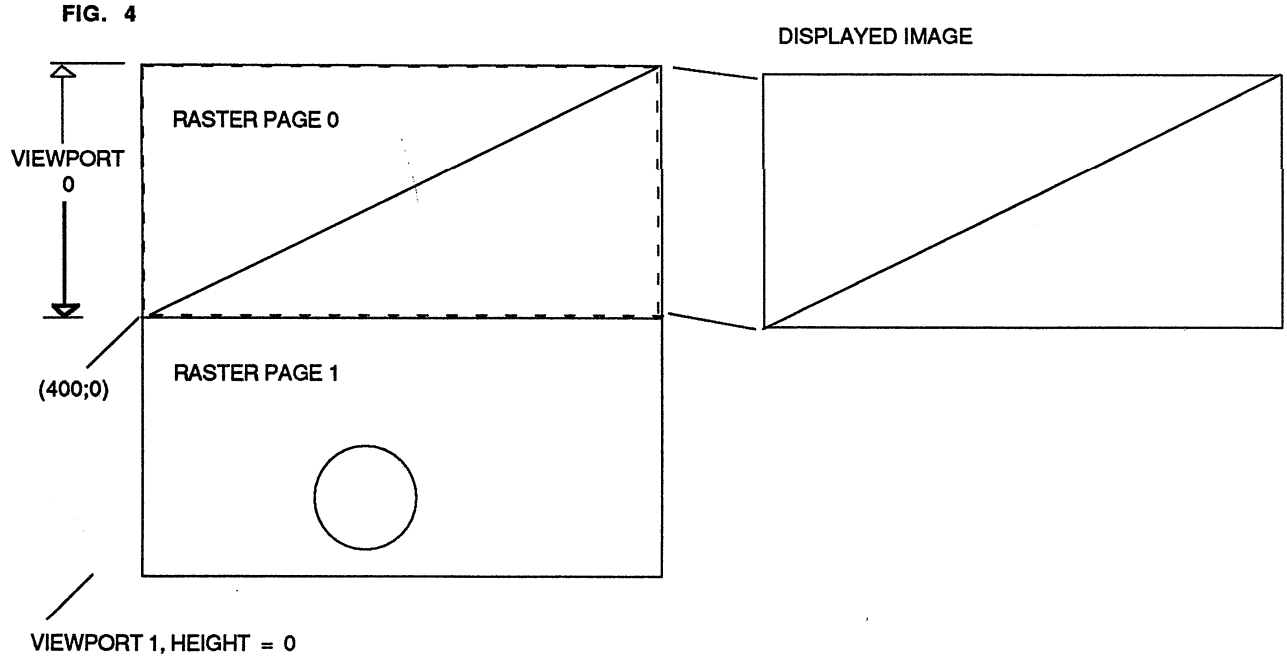
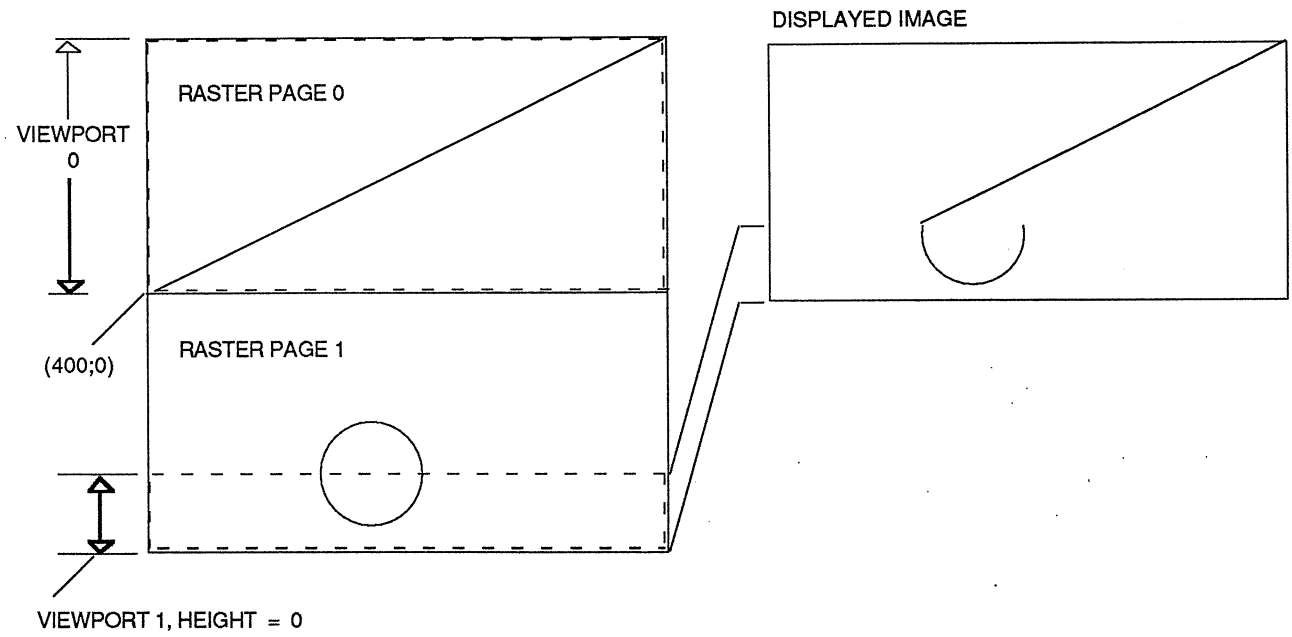


FIG. 5 is identical to FIG. 4 , except that **viewport 1 now has a height of 100 pixels. Raster page 0 is being displayed by viewport 0,** and **the lower 100 pixels of raster page 1 are being displayed by viewport 1**,overlapping that part of viewport 0.

**FIG. 5**




## 4.1 VIEWPORT MODES

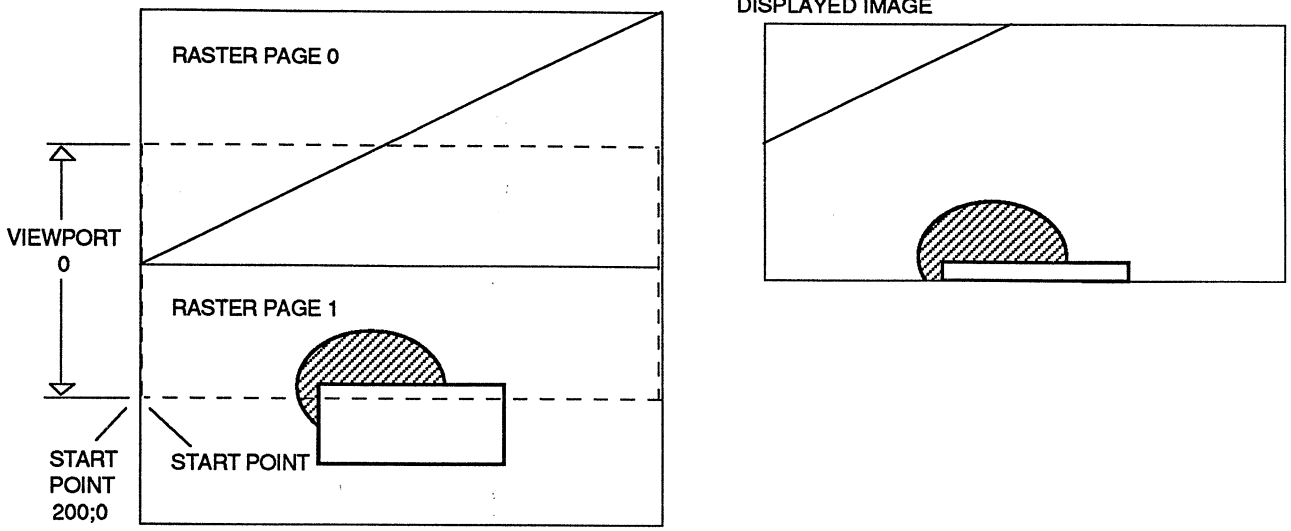
The "Start-Point" (lower left corner) of a viewport is defined by the reference position +Y offset, (an X *offset may be given , but this would produce a distortion in the image being displayed). The reference* postion is defined by the ***viewport mode.***

The modes and their associated reference positions are listed below:

| MODE | REFERENCE POSITION |
|---|---|
| Alpha Absolute | Bottom of Alpha Page (line 24) |
| Alpha Relative | Current text cursor row |
| Graph Absolute | Global Origin (0;0 in Global coordinates) |
| Graph Relative | Graph Window Origin |

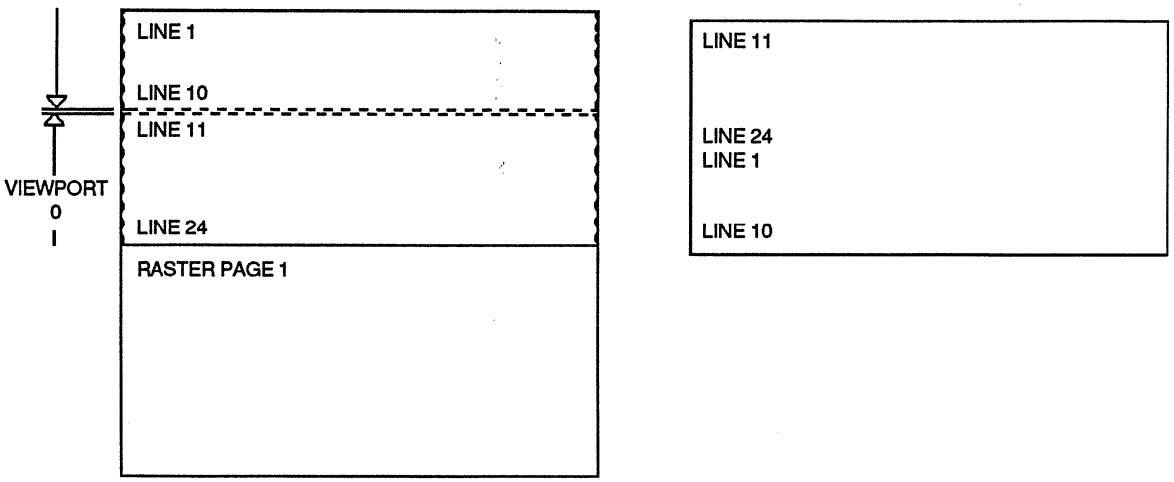
In FIG. 6, viewport 0 is in Graph Absolute Mode, making the reference position the global origin (0;0 in global coordinates). Viewport 0 has been given an offset of 200;0, making the "start position" 200;0 (in global coordinates).

FIG. 6




The circular nature of page 0 must be taken into account when displaying portions of the page within a viewport. This circular nature is envisioned by thinking of the page's top and bottom boundries as being one and the same. The bottom of the page "wraps around" to the top of the page. The result of such a model is illustrated by the defining of a viewport to be Alpha Relative, full screen size, and specifying line 10 as the reference position. The resulting screen image would have lines 1 through 10 at the bottom, and lines 11 through 24 at the top. The two 8 pixel vertical margins would be combined and would be visible between lines 24 and 1. FIG. 7 illustates this case.

FIG. 7




Upon power-up, the viewport default is:

Viewport 0 = Alpha Absolute (displaying raster page 0).
Graphic window origin =400;0 with no offset.

This configuration allows the ST3220 to display alpha text functions as well as graphic objects , using raster page 0. Keep in mind that if the raster page is scrolled, the graphic image will also scroll.

If the on screen keyboard is used, viewport 1 may not be used while the keyboard is on the screen. The keyboard uses viewport 1 to overlay the image on viewport 0.

# INDEX


ACTUATE BUTTON COMMAND 39
ALTERNATE CHARACTER SET 66
ANSI 3.64 DESCRIPTION 7
BORDER SUPPRESS 40
BUTTON AUTO-DRAW 40
BUTTON BEEP 40
BUTTON COMMANDS 36
BUTTON LABEL PACK MODE 40
BUTTON LABELS 38
BUTTON LOCATION, DEFINE 37
BUTTON LOCATIONS AND RESPONSES 81
BUTTON MANIPULATION COMMANDS 39
BUTTON MENU PAGES 41
BUTTON MODES 40
BUTTON PAGES 82
BUTTON RESOURCE INQUIRY 42
BUTTON RESPONSE, DEFINE 37
BUTTON TIME-OUT 44
BUTTON ZERO HIGHLIGHT 40
BUTTONS, DRAWING AND ERASING 41
BUTTONS ENABLED 40
BUTTONS ENTRY/EXIT 40
CLEAR SCREEN ON PAGE CHANGE 40
CLEARING PAGES 42
COMMAND COMPARISON CHART 57
**COMMAND DESCRIPTIONS 17**
- ALPHA CONTROL 20
- AREA FLASH 21
- AUXILIARY MEMORY 20
- CHARACTER 20
- CHARACTER ATTRIBUTES 18
- CHARACTER DEFINITION 20
- CURSOR POSITIONING 18
- DATA/DISPLAY LISTS 20
- EDITING 18
- EIGHT BIT C1 CONTROL CODES 19
- FAST VECTOR 21
- GRAPH WINDOW 20
- GRAPHIC ATTRIBUTES 21
- GRAPHIC OBJECT 20
- OTHER 20

**COMMAND DESCRIPTIONS (Con't.)**
- PRINTING 18, 20
- RASTER OPERATIONS 21
- REPORT 20
- REPORTS 18
- RESET 18
- RESET MODE (ANSI) 19
- RESET MODE (DEC) 19
- SCROLLING 18
- SET MODE (ANSI) 18
- SET MODE (DEC) 18
- SET/RESET MODE 19
- TABS 18
- VIEWPORT CONTROL 20
- VT52 ESCAPE SEQUENCES 19

**COMMAND DESCRIPTIONS, DETAILED**
- ALL ATTRIBUTES OFF 22
- ALPHA CHARACTER SIZE 27
- ALPHA WRITE MODE 27
- ANSI MODE 24, 25
- APPEND DATA LIST 28
- ARC DRAW, ABSOLUTE 30
- ARC DRAW, RELATIVE 30
- AUTO PRINT OFF 23, 25
- AUTO PRINT ON 22, 25
- AUTO REPEAT OFF 24
- AUTO REPEAT ON 24
- AUTO WRAP OFF 24
- AUTO WRAP ON 24
- AUXILIARY MEMORY CHECKSUM REQUEST 27
- AUXILIARY MEMORY CHECKSUM RESPONSE 27
- AUXILIARY MEMORY TYPE 29
- BLOCK FILL DRAW, ABSOLUTE 30
- BLOCK FILL DRAW, RELATIVE 30
- CHARACTER ATTRIBUTE SELECT 31
- CHARACTER SET SELECT 27
- CIRCLE DRAW, ABSOLUTE 29
- CIRCLE DRAW, RELATIVE 29
- CLEAR ALL LISTS 28
- CLEAR ALL TABS 23
- CLEAR GRAPHIC WINDOW 29

**COMMAND DESC., DETAILED (Con't.)**


CLIPPING DISABLE 26
CLIPPING ENABLE 26
CLOSE LIST 28
COMPUTE/STORE CHECKSUM 29
CONCAVE POLYGON FILL, ABSOLUTE 30
CONCAVE POLYGON FILL, RELATIVE 30
CONVEX POLYGON FILL, ABSOLUTE 30
CONVEX POLYGON FILL, RELATIVE 30
CR ONLY 24
CURSOR BLOCK STYLE 26
CURSOR DOWN 21, 25
CURSOR HOME 22, 25
CURSOR LEFT 21, 25
CURSOR POS. REPORT 23
CURSOR POSITION 21
CURSOR RIGHT 21, 25
CURSOR UNDERLINE STYLE 26
CURSOR UP 21, 25
DEFINE ALTERNATE CHARACTER 29
DEFINE VIEWPORT 28
DEFINE WINDOW AREA 27
DEFINE WINDOW ORIGIN 27
DEFINE WINDOW TYPE 28
DEVICE STATUS 23
DIRECT CURSOR ADDRESS 25
ENTER FAST VECTOR MODE 32
ENTER GRAPHICS 25
ERASE GRAPHIC WINDOW 28
ERASE LINE 22
ERASE LINE FROM CURSOR 22
ERASE LINE TO CURSOR 22
ERASE SCREEN 22
ERASE SCRN FROM CURSOR 22
ERASE SCRN TO CURSOR 22
ERASE TO END OF LINE 25
ERASE TO END OF SCRN 25
EXECUTE DISPLAY LIST 28
EXIT GRAPHICS 25
FILL STYLE ATTRIBUTE, SELECT 31
FLASH AREA , START 32
FLASH AREA , STOP 32
FLASH AREA, DEFINE 32
GRAPH CURSOR BOX TYPE 26
GRAPH CURSOR CROSS (X) TYPE 26
GRAPH CURSOR OFF 26
GRAPH CURSOR ON 26
GRAPH STRING CONTROL OFF 26
GRAPH STRING CONTROL ON 26
GRAPH STRING NO-WRAP 26
GRAPH STRING WRAP 26
GRAPH WRITE MODE SELECT 31
GRAPHIC STRING DRAW, ABSOLUTE 31
GRAPHIC STRING DRAW, RELATIVE 31
HORIZ TAB CLEAR 23
HORIZ TAB SET 23
HORIZ. & VERT. POSITION 22
IDENTIFY TERMINAL 25
INDEX 22
LINE STYLE ATTRIBUTE, SELECT 31
LIST REPORT 28
LOCATE VIEWPORT 28
LOCK KEYBOARD 23
MULTIPLE ATTRIB. ON/OFF 22
NEWLINE 23
NEXT LINE 22
NO PRINT FORMFEED 24
NORMAL VIDEO (SCREEN) 24
OPEN DISPLAY/DATA LIST 28
PRINT BUFFER REQUEST 26
PRINT BUFFER RESPONSE 26
PRINT CONTROL OFF 23, 25
PRINT CONTROL ON 23, 25
PRINT CURSOR LINE 23, 25
PRINT EXTENT=SCREEN 24
PRINT EXTENT=SCROLL 24
PRINT FORMFEED 24
PRINT INPUT BUFFER 26
PRINT INPUT DISABLE 26
PRINT INPUT ENABLE 26
PRINT INPUT NO BUFFER 26
PRINT SCREEN 23, 25
PRINTER PORT CONFIGURE 27
PRINTER STATUS 23
RASTER MEMORY READ REQUEST, ABSOLUTE 32
RASTER MEMORY READ REQUEST, RELATIVE 32
RASTER MEMORY READ RESPONSE 32
RASTER MEMORY WRITE , RELATIVE 32
RASTER MEMORY WRITE, ABSOLUTE 31
READ DATA/DISPLAY LIST 28
RESET TO INITIAL STATE 23

**COMMAND DESC., DETAILED (Con't.)**
REVERSE INDEX 22
REVERSE LINEFEED 25
REVERSE VIDEO (SCREEN) 24
REVERSE VIDEO OFF 22
REVERSE VIDEO ON 22
SCREEN SAVER OFF 26
SCREEN SAVER ON 26
SCREEN SAVER TIMEOUT 29
SCROLL DOWN 21
SCROLL MARGINS 21
SCROLL UP 21
SET GRAPHIC CURSOR 29
SET-UP KEY DISABLED 26
SET-UP KEY ENABLED (DEFAULT) 26
ST3220 SYSTEM STATUS REQUEST 27
ST3220 SYSTEM STATUS RESPONSE 27
STRING ATTRIBUTE SELECT 31
TEXT CURSOR OFF 24
TEXT CURSOR ON 24
TRANSMIT 7-BIT EQUIV 24
TRANSMIT 8-BIT CNTRL 24
UNLOCK KEYBOARD 24
VECTOR DRAW, ABSOLUTE 29
VECTOR DRAW, RELATIVE 29
VT52 MODE 24
XON/XOFF DISABLED 26
XON/XOFF ENABLED (DEFAULT) 26
COMMAND SUMMARY 61
CONDUIT COVERS, INSTALLATION PROCEDURES 77
CONFIGURATION INQUIRY 46
**CONNECTOR PIN ASSIGNMENTS**
HOST INTERFACE CONNECTION 51
KEYBOARD INTERFACE CONNECTION 52
PRINTER INTERFACE CONNECTION 52
COPYING RESPONSES 38
CUSTOMER SUPPORT 79
DEFAULT SETTINGS 55
DRAWING OBJECTS 13
ERASING OBJECTS 14
FAST VECTOR MODE 15
GRAPHIC TERMINAL OPERATION 9
GRAPHIC TEXT 14
GRAPHIC WINDOW 83
HIGHLIGHT BUTTON TOUCHES 40
INSTALLATION 1
KEYBOARD 71
KEYBOARD, CONNECTING TO 1
KEYBOARD MODES 44
LOCAL RESPONSES 38, 81
MECHANICAL DRAWINGS 53
MODEM, CONNECTING TO 1
MULTI-STATE BUTTONS 39
ON-SCREEN KEYBOARD 44
POLYGON FILL 15
POP-UP MENUS 43, 82
POWER-UP MESSAGES 73
PRINTER, CONNECTING TO 1
RASTER (VIDEO) MEMORY 83
RETAINED GRAPHIC FRAME STORAGE 65
RS-232C INTERFACE 67
RS-485 MULTI-DROP 47
SCREEN VIEWPORTS 10
SELECTING A PAGE 41
SELF-CHECK FUNCTIONS 45
SET BUTTON STATE 40
SET-UP, ENTERING AND LEAVING 3
SET-UP MENUS 3
SET-UP MENUS, COMMUNICATIONS 4
SET-UP MENUS, DISPLAY 4
SET-UP MENUS, GENERAL 3
SET-UP MENUS, KEYBOARD 4
SET-UP MENUS, PRINTER 4
SET-UP MENUS, TOUCH 5
SOFTWARE COMMANDS 34
SPECIFICATIONS 49
ST3225 75
ST3420, MECHANICAL DRAWINGS 54
TEXT TERMINAL OPERATION 8
TOUCH CAPABILITIES 33
TOUCH INQUIRY 36
TOUCH MAINTENANCE 33
TOUCH PROTECT 45
TOUCH REPORTS 35
TOUCH SPECIFICATIONS 33
VIEWPORT MODES 86
VIEWPORTS 84
VIEWPORTS, SCREEN 10

SealTouch Graphics Terminal
(Model ST3220)
Manual
Ver. 1, Rev. B
September 1, 1990
Part No. 8613
Printed in U.S.A.
All Rights Reserved

**LUCAS DEECO**
31047 Genstar Road
Hayward, California 94544-7831
(510) 471-4700
Fax: (510) 489-3500